

- ☐ **Assembler: acceso a disco desde residentes**
- ☐ **Experimentando con su PC: Medición acústica de distancias**

PC Práctica

publicación mensual de Editorial Cul - Tec S. A.

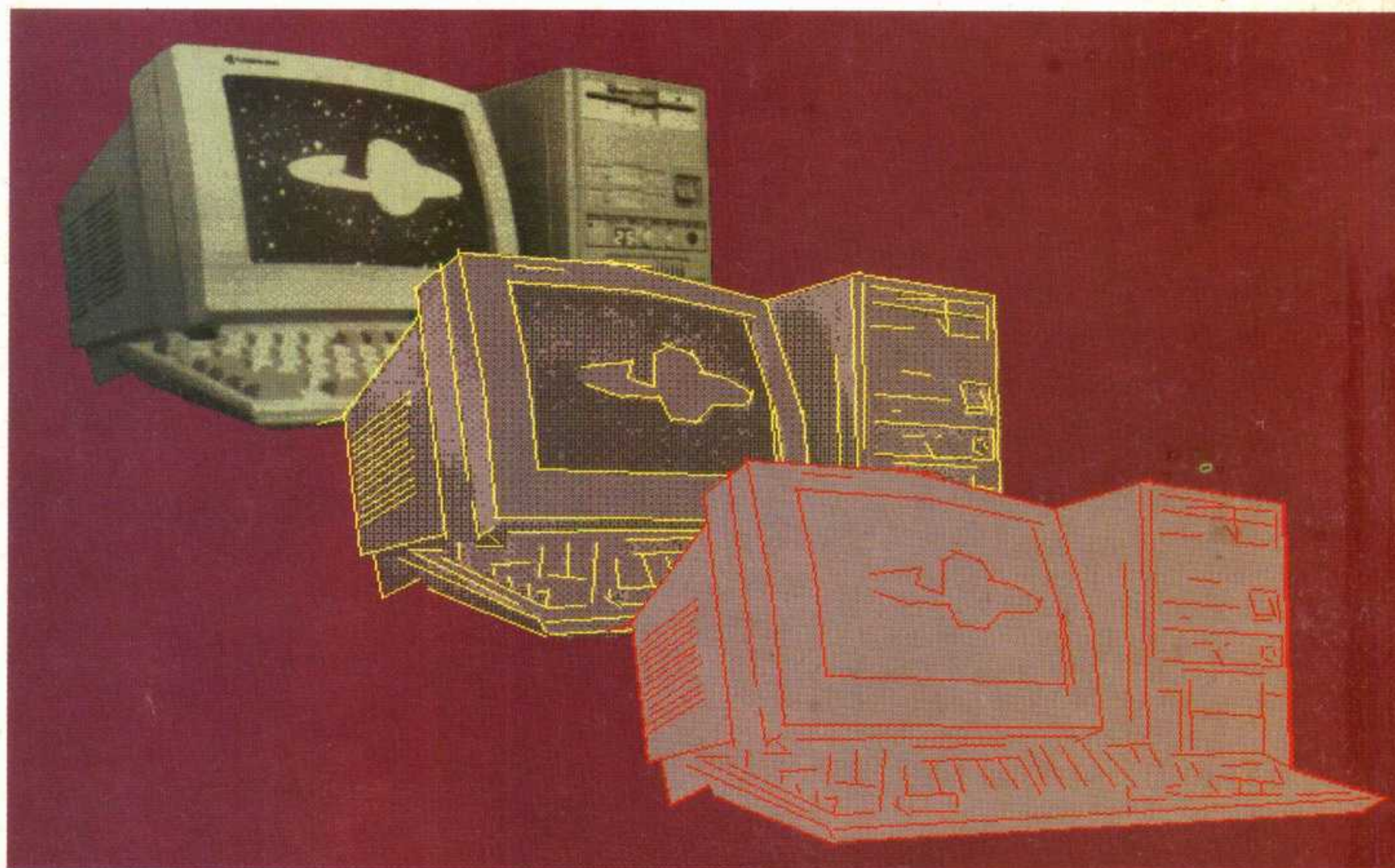
Año 4 nº 35

Marzo de 1995

ISSN 0327 - 6600

\$ 4.-

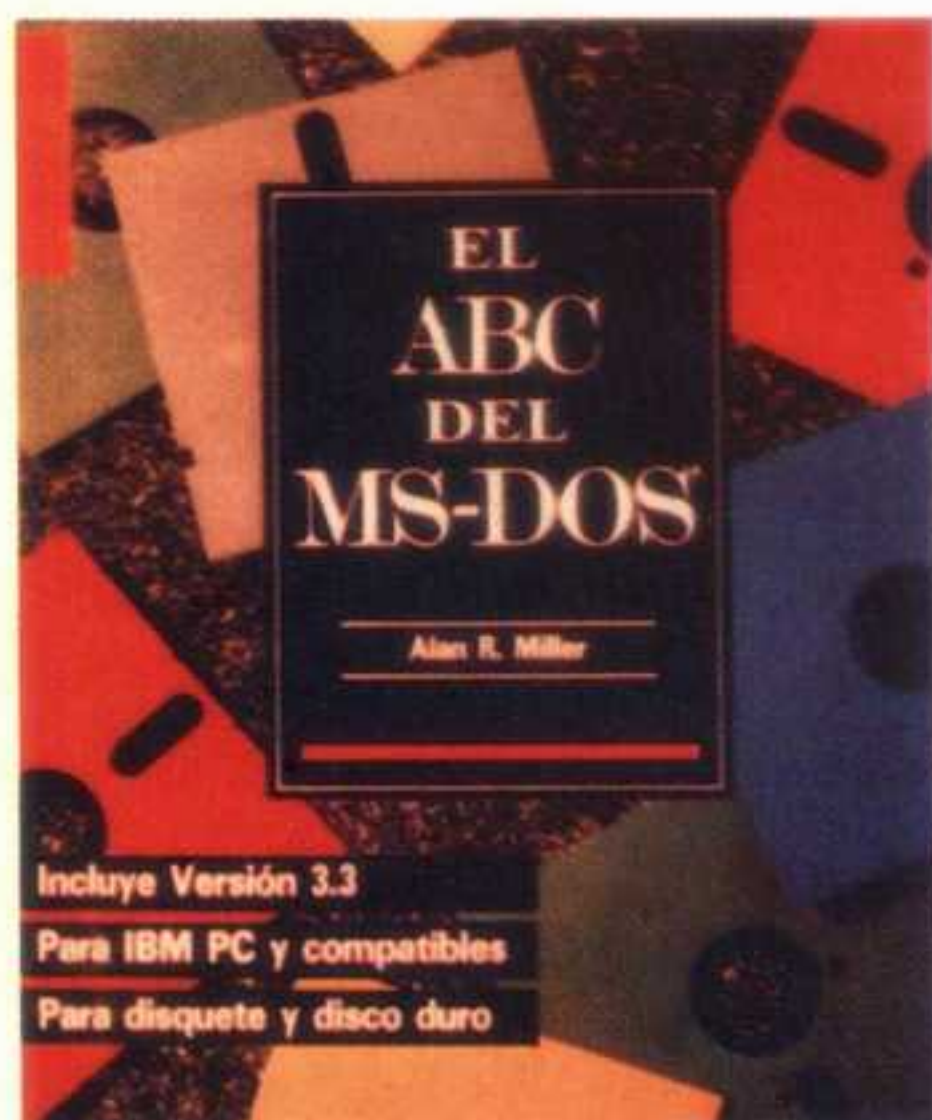
PATRONES



*Método de
reconocimiento*

- ☐ **Técnicas gráficas: Medios tonos**
- ☐ **Arquitectura de la PC: RAM y ROM**

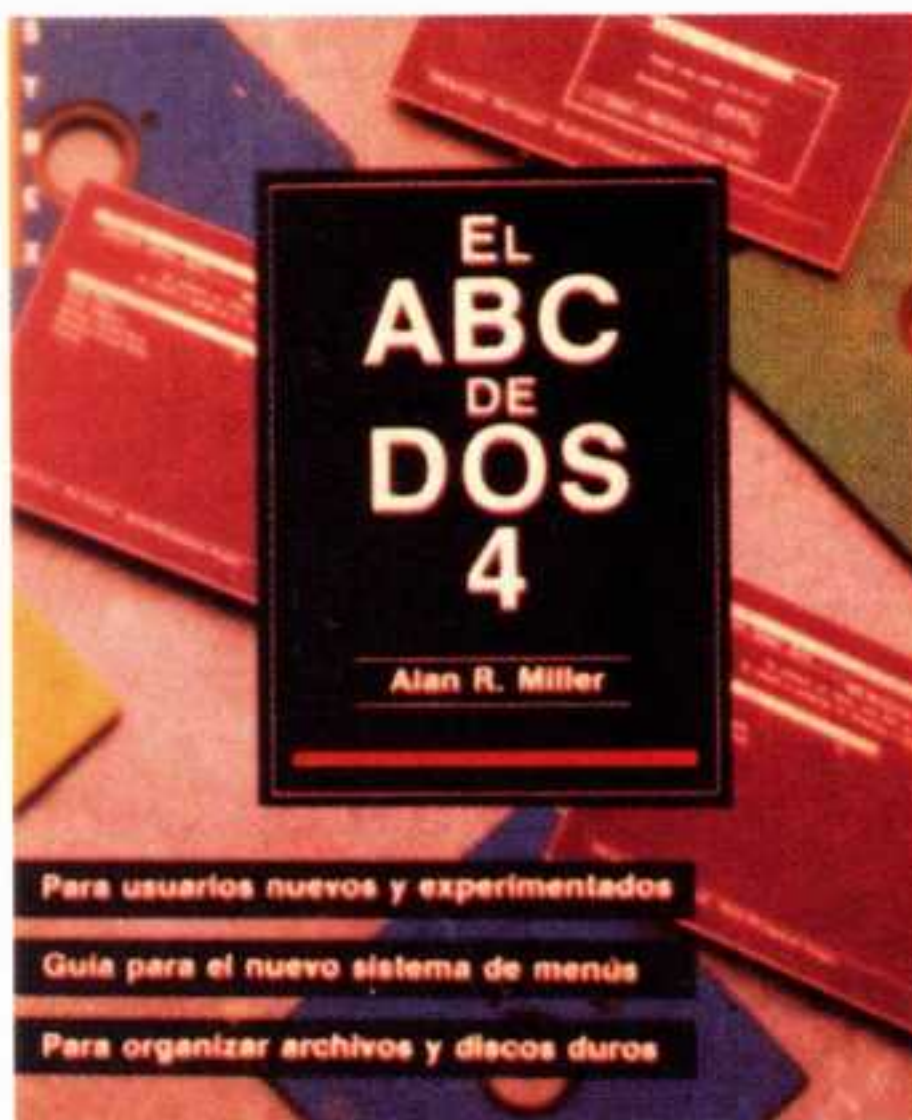
el ABC de la informática...



El ABC del MS-DOS

Manual claro y completo destinado a quienes desean conocer y aprovechar el poder de las PC. Una guía progresiva referente a todos los aspectos del Sistema Operativo.

\$ 27.-



El ABC del DOS 4

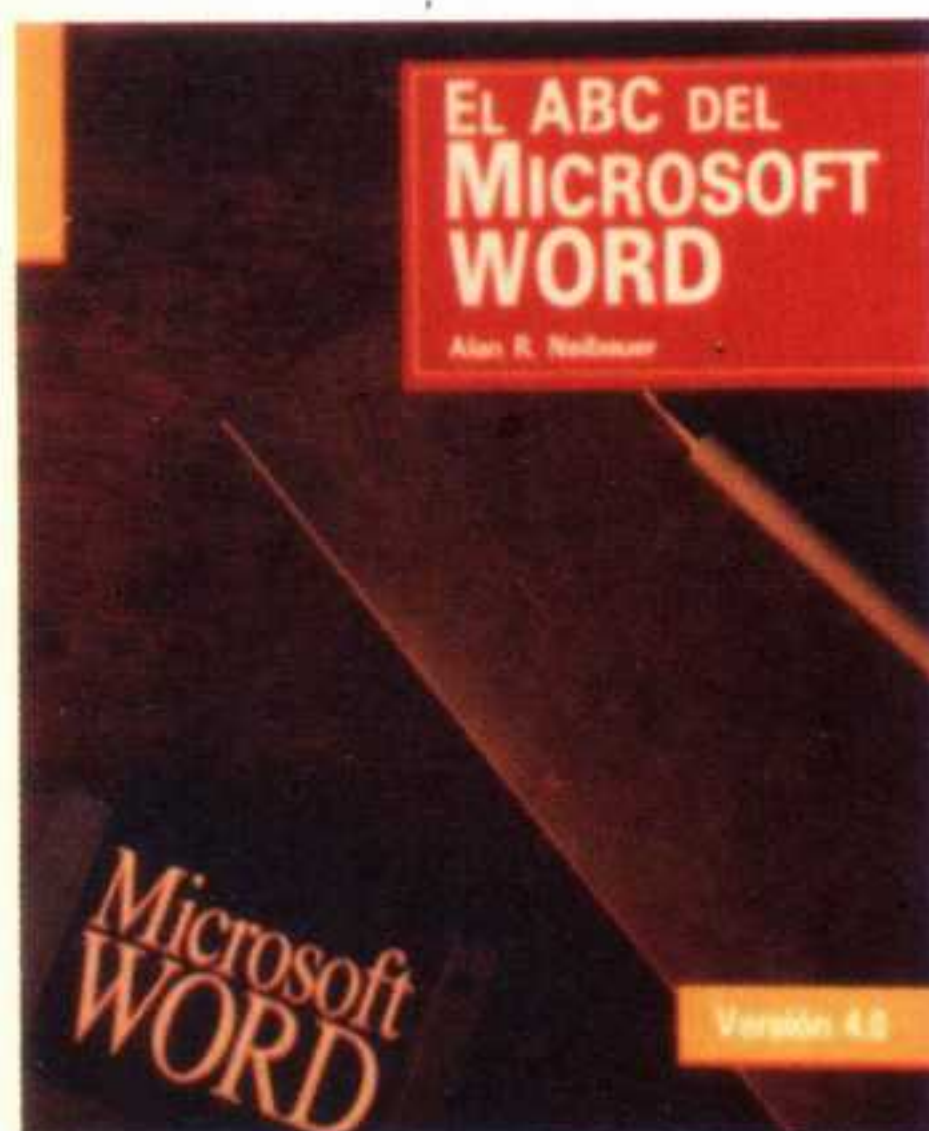
Una guía práctica y sencilla para el uso de DOS versión 4 en una IBM PC, XT, AT, PS/2 o sistema compatible. La claridad de las instrucciones le permitirá el manejo eficiente del sistema.

\$ 29.-



El abecé de MS-DOS 5

Este libro está organizado para que un principiante aprenda a utilizar el sistema operativo MS-DOS 5. También, pensando en los profesionales, se exponen órdenes y tareas más complejas.



El ABC del MS WORD

Le permitirá utilizar rápidamente el programa. Con los ejemplos prácticos que se brindan, no tardará en adquirir el dominio necesario para emplear este poderoso procesador de texto.

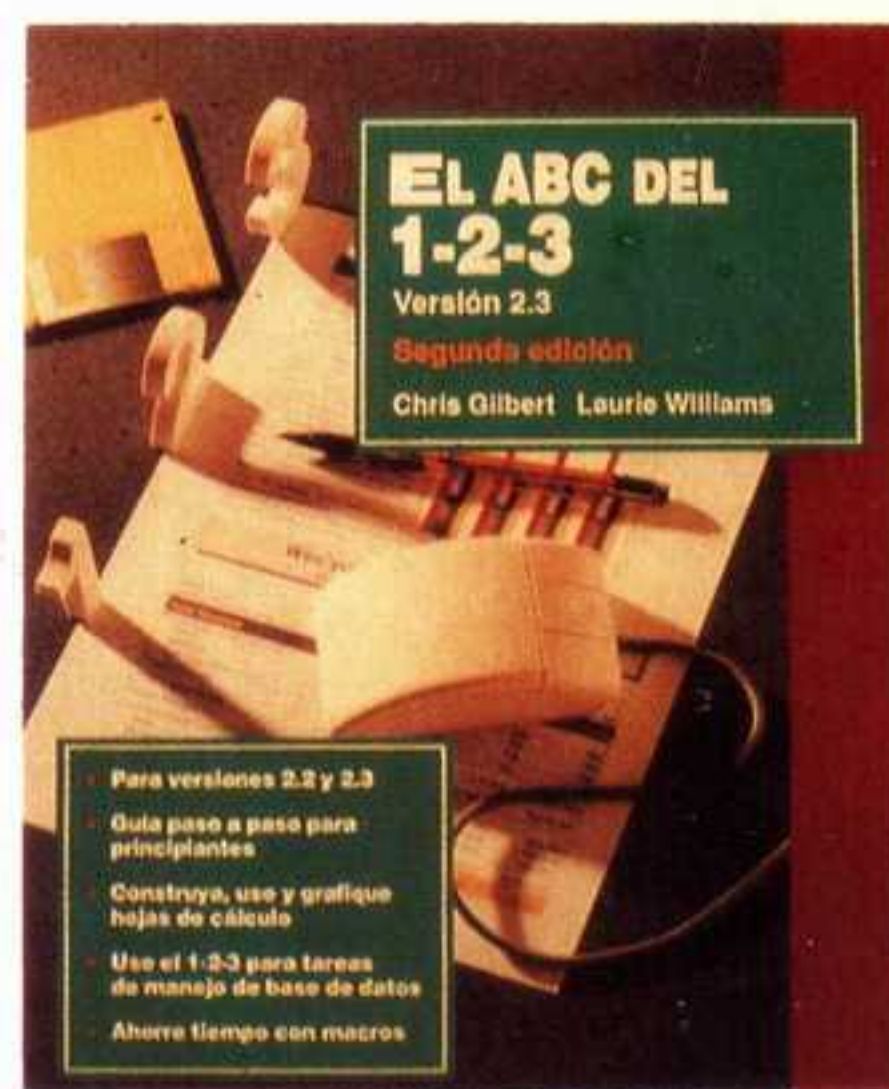
\$ 27.-



El ABC del dBASE IV 1.1

Escrito en lenguaje cotidiano, fácil de seguir, tiende a obtener rápidamente resultados prácticos. Siguiendo las instrucciones, podrá aprender a crear y editar bases de datos.

\$ 28,60



El ABC del 1-2-3

Todas las bases del Lotus 1-2-3 versión 2.2 y 2.3 en una guía fácil de usar y consultar. El ABC del 1-2-3 versión 2.2 y 2.3 es un tutorial práctico escrito especialmente para principiantes.

\$ 28.-

Solicítelos agregando \$ 6.- para envío por Correo Certificado, mediante giro postal o bancario a la orden de:

EDITORIAL CUL-TEC S.A.

Independencia 1654 - 1100 Buenos Aires

Tel /Fax 383-7126 381-9308 381-9327

Horario: Lunes a Viernes de 10 a 17 horas



PC Práctica

u m a r i o

Publicación mensual de

editorial cul - tec s.a.

Presidente

Juan S. Cutrone

Vicepresidente

Pablo R. Mazzitelli

Av. Independencia 1654

1100 - Buenos Aires

Tel/Fax 383-7126 381-9308/9327

Director

Ing. Victor O. Cutrone

Director Técnico

Lic. Fernanda A. Tassa

Jefe de Redacción

Ing. Carlos A. Serventi

Publicidad

Arg. Silvia M. Catanese

Diseño y diagramación

Juan J. Vicente

CORREO ARGENTINO CENTRAL (B)	FRANQUEO A PAGAR
	CUENTA N° 1398
	TARIFA REDUCIDA CONCESION N° 2078

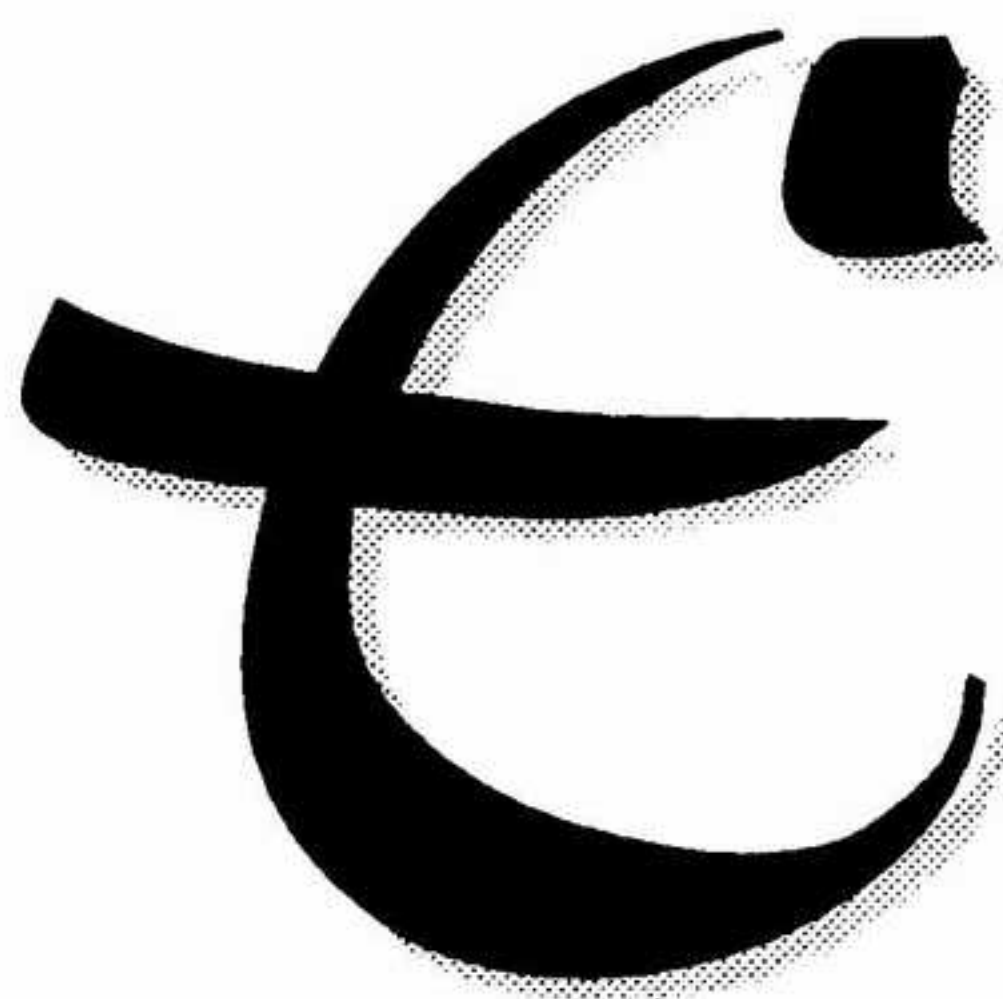


Impreso en
Talleres Gráficos Conforti SA.
Av. Reg. de los Patricios 1941 Cap. Fed

• Editorial	4
• Arquitectura de la PC RAM y ROM	5
• Reconocimiento de patrones Determinación del ángulo de "skew"	10
• Experimentando con su PC Medición acústica de distancias	24
• Novedades en shareware Basics & Beyond	30
• Técnicas gráficas Medios tonos	32
• Assembler para todos Acceso a disco desde TSRs	40
• Desafío La araña y la mosca	46
• Informática y ecología El ahorro energético	48
• Correo de lectores	50

Dirección Nac. del Derecho de Autor N° 368266
Reg. de Pat. y Marcas Tit. de Prop. A 1815464
Prohibida la reproducción total o parcial
sin autorización expresa
La responsabilidad de los artículos firmados co-
rresponde únicamente a los autores
Miembro de Asoc. Arg. de Editores de Revistas
Asoc. de la Prensa Técnica Arg.

Concesionarios para la venta
en la Capital y Gran Buenos Aires
Distribuidora Cancellaro S.R.L.
V. del Pino 2639 5ªA - Buenos Aires
Tel. 782-7925
en el interior del País
Distribuidora Bertran S.A.C.
Vélez Sarsfield 1950
Buenos Aires



ditorial

En marzo de 1992 -coincidiendo con el nacimiento de PC Práctica- Michelángelo acaparó la atención periodística después de haberse presentado en sociedad arrasando con los discos duros de muchos imprecavidos. Nadie quedó entonces inadvertido de la existencia de los virus informáticos, que así se incorporaron a nuestro folclore con esa deliciosa superficialidad de nuestro saber popular.

Hoy, tres años después, han prosperado las vacunas, y ya no hay quien no tenga una clínica entera en su PC; hasta se extraña que la suspicacia argentina no haya imaginado mitos paralelos a los atribuidos a los fabricantes de pediculicidas, que, como todo el mundo sabe, "*siembran piojos en los colegios*". Pero, bromas aparte, poco es lo que hemos aprendido.

Luego de tres años de buen comportamiento (debido sobre todo a que el 6 de marzo fue feriado tanto el año pasado como el anterior), *hemos decidido olvidarlo*. Como consecuencia, el pasado lunes 6, muchos encontraron sus discos rígidos prolijamente destruidos, en lo que a información se refiere.

No practicamos la memoria. Sabemos que el tiempo cura todos los males, y comenzamos a intuir que en Argentina perdona también todos los pecados. Y el caso me suena a conocido. *Pero no puedo recordar por qué.*

La Dirección



RAM y ROM

Comenzamos el análisis de los subsistemas de RAM y de ROM



Nuestra nota anterior presentó breve y genéricamente las DRAMs; a ese esquema general corresponden multitud de dispositivos. En esta oportunidad comenzaremos por ver cómo se organizan DRAMs y ROMs típicas en un sistema PC.

La descripción referida de las RAMs dinámicas se basa en memorias de Nx1 bit (N celdas totales, y un "bus" de datos de 1 bit por chip; esto implica 8 chips para sistemas de 8 bits, 16 para sistemas de otros tantos, etc.), un número indeterminado de líneas de dirección, y un par de líneas de validación (lectura y escritura).

Para retomar adecuadamente el hilo de esa descripción, recordemos el papel de dos señales características de las

DRAMs, las **validaciones** de fila y columna **RAS** y **CAS**. Estas permiten reducir la cuenta de terminales necesarios multiplexando el bus de direcciones, las que terminan siendo suministradas en dos etapas, requiriendo así la mitad de líneas.

En un chip de RAM dinámica concreto aparecen algunas variantes. Dependiendo de la cantidad de celdas, se tendrá un cierto número **n** de líneas de dirección, tal que:

$$C \leq 2^{2n+R}$$

R representa la posibilidad de codificar uno o dos bits de dirección adicionales, no a través de las líneas de dirección, sino por medio de un esquema de RAS Y CAS más complejo, por ejemplo dos señales de RAS independientes.

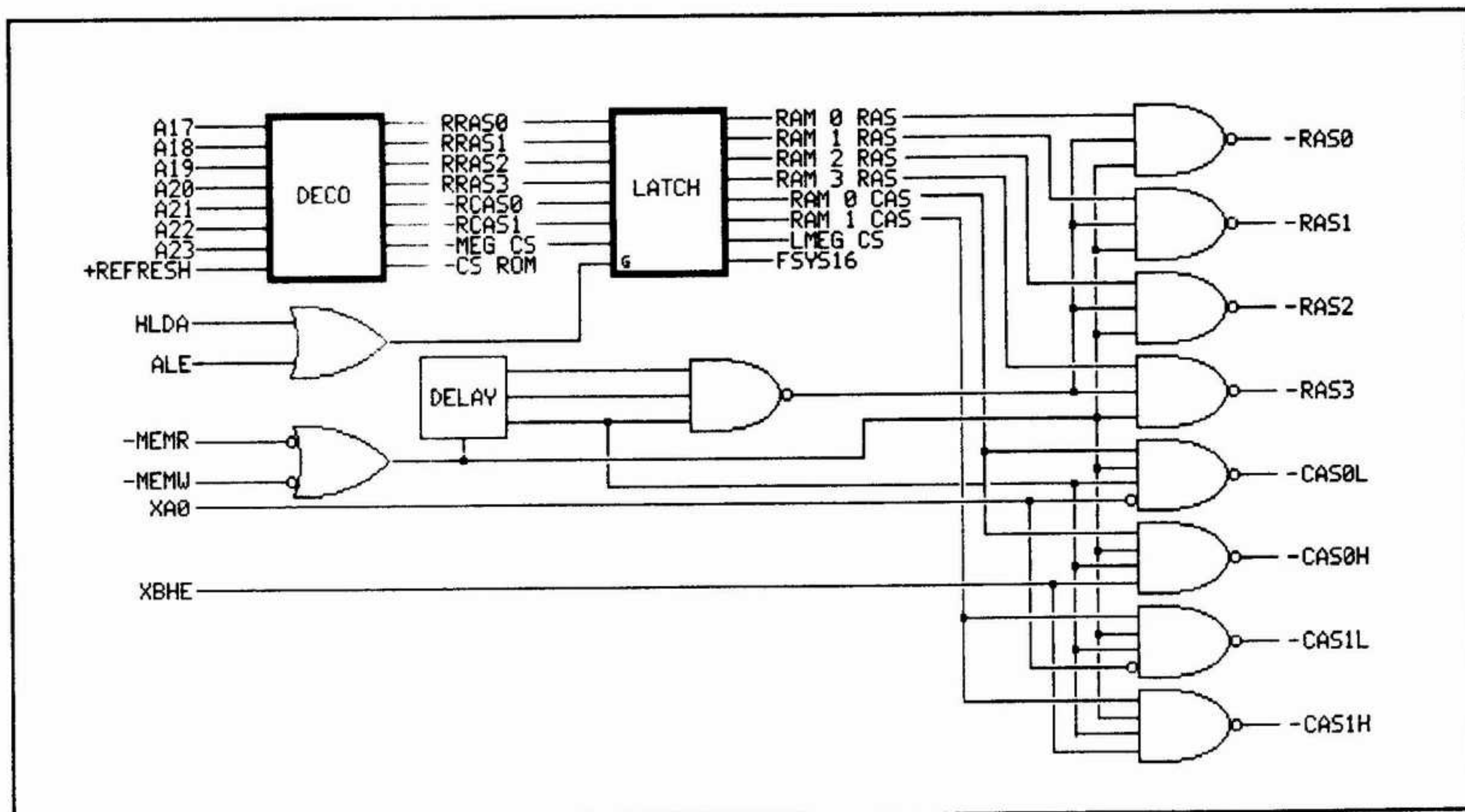


Fig. 1 - Esquema de generación de las señales de validación RAS y CAS

Habilitación de lectura y escritura

Consideradas en nuestra nota anterior como señales independientes, se presentan de forma ligeramente distinta: selección de chip y validación de escritura. La primera determina el acceso, sea en lectura o escritura. Cuando la señal es válida, el chip queda seleccionado. La segunda define si el acceso es de lectura o de escritura.

La selección de chip puede no requerir una línea físicamente separada, tomándose de la combinación de RAS y CAS.

Ancho de bus

Hasta aquí consideramos, por simplicidad, memorias de 1 bit, pero se presentan chips de di-

ferentes cantidades de bits, que corresponden a conjuntos de otros tantos bancos de 1 bit "en paralelo". Es decir, con señales de dirección y validación comunes, y una cantidad correspondiente de líneas de datos, generalmente 2 o 4.

Módulos

La diversidad recién expuesta (cantidad de líneas de dirección, datos y validación de fila RAS) hace que un sistema sólo acepte un modelo de RAM de los numerosos existentes; si además se prevén diferentes configuraciones en lo que a cantidad de RAM se refiere, se requerirán jumpers de configuración. El resultado -incluyendo la complicación adicional de los chips de paridad- es un soporte

de RAM que para 1 Mb ya resulta físicamente complejo.

Esto favoreció el empleo de **módulos**, que simplemente son pequeñas placas con varios chips de RAM y configuración de terminales standard. De este modo, módulos constituidos por chips diferentes pueden presentar un mismo pinout standard. El módulo reúne suficientes chips como para presentar una capacidad de 8 bits x N KB+ 1 bit x N Kb de paridad, y permite además la **detección automática**: dos o tres terminales se emplean exclusivamente para indicar la presencia del módulo y su tipo.

En una época se popularizaron los módulos **SIP (Single In-Line Package**, empaque de terminales en una línea), abando-

nados rápidamente en favor de las placas **SIMM (Single In-line Memory Module)**, más fáciles de insertar, aunque de contactos menos seguros.

Para ejemplificar la estructura de un módulo, consideremos un SIMM de 1MB x 8bits. Una combinación típica para él es: 2 chips de 1Mb x 4 bits + 1 chip de 1Mb x 1 bit (paridad).

Ahora puede deducir el lector de dónde proceden las restricciones en cuanto a configuración de memoria. Tomemos como ejemplo un sistema 386DX. El bus es de 32 bits; cada módulo de 8. Es evidente que se necesitan 4 módulos "en paralelo" para configurar N celdas de memoria, con N=1, 2, 4, 8, 16 Mb. Una 386SX (16 bits) requeriría sólo pares de módulos.

Cada conjunto es un **banco**, y las motherboard suelen incluir dos (esto es una cuestión de diseño procedente de la limitación de espacio físico; bien podrían ser cuatro). La consecuencia es la necesidad de poblar cada banco con 4 (o 2, para la SX) SIMMs iguales, y la posibilidad de completar 1, 1+1, 2, 2+1, 2+2, 4, 4+1 Mb, etc.

Bancos y Chips discretos

Agrupamientos similares, menos evidentes, se dan en los sistemas que emplean chips discretos: también en este caso se pueden identificar bancos de N Kb, en cada uno de los cuales se pueden considerar grupos de 8 bits + 1 bit de paridad.

Bytes y paridad

Estos grupos corresponden a los bytes de datos que componen el bus. Consideremos SIMMs de 1Mb x 8bits. Para un banco dado, el primer SIMM corresponde al byte más bajo. Las líneas de datos D0 a D7 del chip corresponden a MD0-MD7 del bus. Para el segundo chip, D0 a D7 son MD8-MD15 del bus (el segundo byte), y así.

La **paridad**, empleada para validar la integridad del contenido de memoria es independiente para cada byte. Si nuestro sistema falla, y disponemos de reporte acerca de la dirección que produjo el fallo, ella será múltiplo de 4 si falló el SIMM del byte menos significativo, múltiplo de 4+1 si el siguiente, y así.

Direccionamiento

Considerando que en un sistema de 32 bits se dispone de conjuntos separados de chips para cada uno de los cuatro bytes del bus de datos, se ve que las dos líneas de dirección más bajas *no forman parte de la dirección del módulo o conjunto*. En efecto, si se lee la dirección 12340, el primer módulo proporcionará el byte de dirección 12340, el siguiente el de dirección 12341, el siguiente, 12342 y el último 12343. Las líneas de dirección 0 y 1 se usan para discriminar (mediante CAS separados) *qué módulos se leen*; las líneas 2, 3, 4 corresponden a las direcciones 0, 1, 2 del chip, cuando se está codifi-

cando la parte baja de la dirección, y, por ejemplo, (porque esto depende de la capacidad del chip) a las líneas 8, 9, 10, etc. cuando se codifica la parte alta.

Hasta ahora estamos considerando un sistema de direccionamiento **fijo**: es posible identificar direcciones de los chips con direcciones del sistema, en plena correspondencia. ¿Qué ocurriría si pudiéramos "cruzar cables" o negar líneas de dirección? Por ejemplo, podríamos hacer que los que en el sistema constituye la dirección **Fxxxx** para los chips de memoria apareciera como **Dxxxx**.

La consecuencia sería un **re-mapeo** de la RAM, que aparecería entonces ocupando espacios diferentes de los originales. En el ejemplo citado, parte de la RAM aparecería en el espacio de memoria tradicionalmente reservado para la ROM del sistema.

Los chipsets actuales implementan estas variantes dinámicamente de manera que es posible redefinir ampliamente la forma de direccionamiento; una de sus aplicaciones es la **shadow RAM**: RAM mapeada en las direcciones correspondientes a una ROM, en la que previamente se copió el contenido de esa ROM. La ventaja: tiempo de acceso reducido. La desventaja: *la RAM es corruptible*.

El lector habrá notado que en algunas ocasiones, luego de uno de esos cuelgues "mach-

zos", un reset software no soluciona los problemas, e incluso presenta indicios claros de que "algo anda mal en el ROM BIOS". Lo que sucede en estos casos es que *ya no hay un ROM BIOS*, sino un BIOS en una RAM corrompida, que no se ha recargado, como habría ocurrido si se hubiera forzado un reset hardware.

ROM

Abandonamos momentáneamente las RAMs para una breve consideración de los componentes ROM del sistema, lo que permitirá tratar en conjunto los aspectos comunes.

Como el lector sabe, se trata de memorias **de sólo lectura**, y su organización es mucho más sencilla, debido a este hecho y a que la capacidad de memoria es *considerablemente menor*. En efecto, 64Kb (típicos) son direccionables con comodidad sin tener que recurrir a multiplexado.

El chip de ROM presenta entonces (*para regocijo del lector*) un tradicional bus de datos de 8 o 16 bits y un sencillo bus de direcciones de 15 o 16 líneas. La selección de chip y la lectura se determinan por medio de un par de líneas de control, generalmente cableadas juntas.

Un subsistema típico de ROM incluye dos chips de 32Kb, que corresponden al byte par (EVEN o LOW) e impar (ODD o HIGH) del bus de datos del sistema. La dirección 0 selecciona cual de ambos chips se lee, las

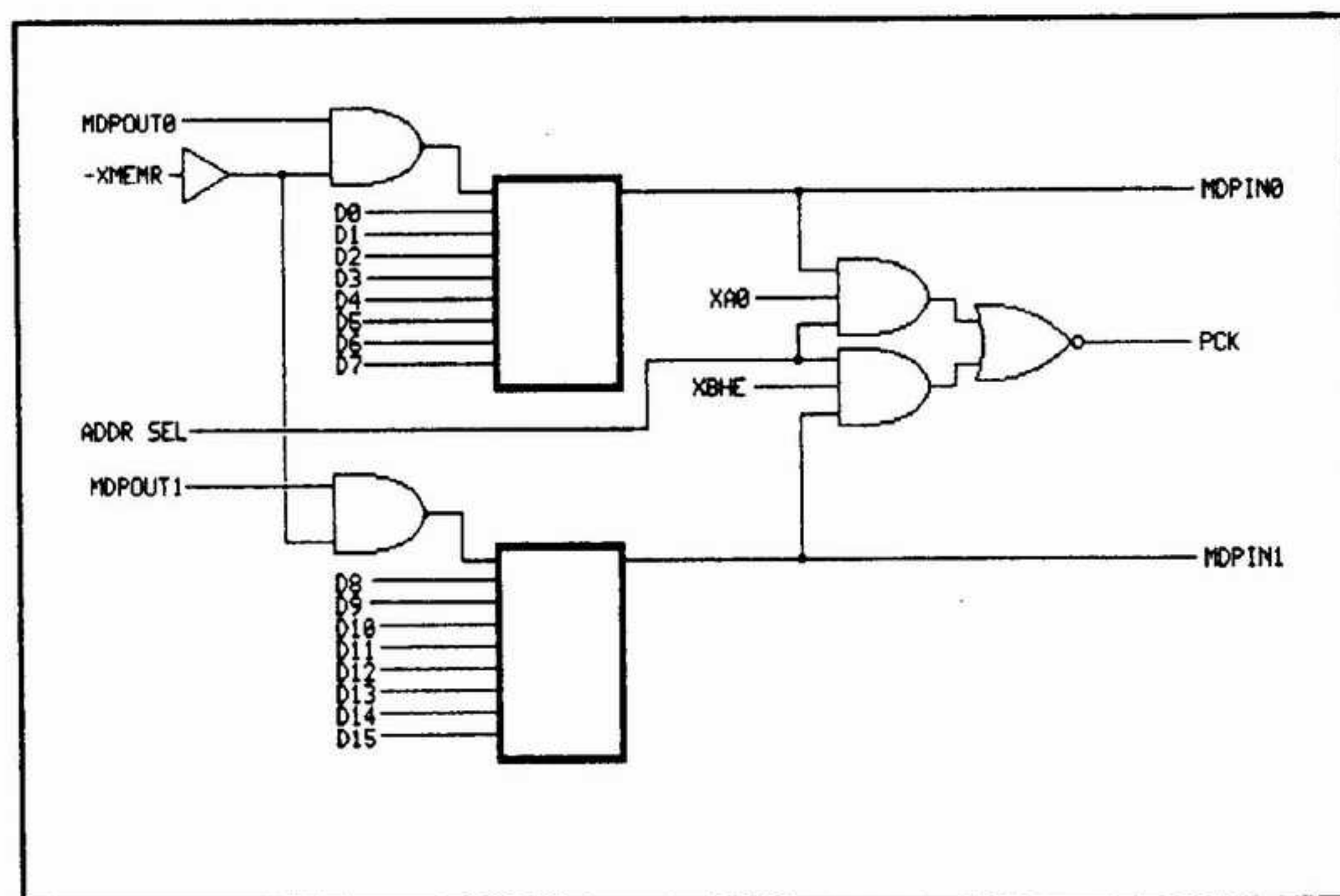


Fig. 2 - Control de paridad (ejemplificado para 16 bits)

1 a 15 fijan las direcciones internas 0 a 14 del chip.

También en un sistema tradicional se prevé un segundo juego de ROMs, mapeado en E000, normalmente no usado.

El subsistema de memoria

Retornando a la RAM, el subsistema de memoria completo incluye:

- Generación de las señales RAS y CAS
- Multiplexado de dirección
- Validación de paridad
- Buffering de direcciones y datos
- Mecanismo de refresco

Generación de RAS y CAS

Por lejos, la mayor complejidad reside en los dos primeros puntos; además, el esquema de generación de RAS y CAS es sumamente dependiente del sis-

tema que se trate. Por estas razones, vamos a describirlo somera y esquemáticamente.

Las señales de validación son deducidas de las **señales altas de dirección**, de la dirección 0 y de algunas señales de control (por ejemplo XHBE, habilitación de la parte alta del bus, XMEMR y XMEMW, lectura y escritura de memoria), y de señales de timing.

Podríamos considerar dos módulos bastante diferentes para cumplir esta función. El primero es un **decodificador** considerablemente complejo, que proporciona en base a las direcciones altas y otras señales de control, los estados que servirán para controlar la RAM, básicamente, varios "juegos" de RAS y CAS. Esto ocurre una vez por ciclo, y esos estados permanecerán **fijos** (*latcheados*) durante todo el ciclo.

Pero hemos visto que la RAM se accede a través de una secuencia relativamente compleja, con distintas señales activas en las diferentes fases de un ciclo.

Así, el segundo módulo se encarga de **temporizar** estas señales, y, por así decirlo, aplicarlas en la secuencia y oportunidad correctas a la RAM.

La primera etapa consta -en este modelo conceptual- de un decodificador y un latch. A partir de la dirección 17, las direcciones se decodifican generando las señales primitivas de RAS y CAS (RRAS y RCAS). Típicamente, se generán 1 o 2 señales de RAS y una señal de CAS para cada banco de memoria. De estas últimas, sólo una por vez resulta activa.

Se aprovecha la decodificación de las líneas altas de dirección para generar, además, -CS ROM (chip select de ROM), -MEG CS (dirección en el primer megabyte, que se emplea para obtener compatibilidad con adaptadores periféricos estilo XT, los cuales están diseñados suponiendo un único Mb de espacio) y FSYS16 (transferencia en 16 bits).

Estas señales son latcheadas sobre el flanco ascendente de ALE, y permanecen durante un ciclo completo.

La etapa restante -de temporización- parte de un OR entre las negadas de MEMR y MEMW, es decir, de una señal

que se activa en cada acceso a memoria. Este pulso es **demorado** en diferentes lapsos, y la señales demoradas se combinan con las RAS y CAS latcheadas en la etapa anterior, proporcionando señales activas en momentos específicos para producir la temporización adecuada. En este proceso intervienen, además las direcciones bajas 0 o 0 y 1 para desdoblarse cada CAS en tantas señales como bytes integran el bus.

Multiplexado

La etapa anterior genera, adicionalmente, la señal que determina cómo se multiplexarán las líneas de dirección (ADDRSEL). El multiplexado consiste, entonces, en MUXes digitales que en base a ADDRSEL conducen la parte baja o la parte alta de las líneas de dirección a los addresses de la RAM.

Paridad

Para el control de *paridad* se emplean circuitos de generación de paridad, a razón de uno por cada byte de datos del bus. Cada circuito recibe 9 bits como input y proporciona una salida de un bit que codifica la paridad de las nueve entradas. Si el número de 1s es par, la salida queda en cero.

Durante las **escrituras** en memoria, 8 de las entradas corresponden a los bits escritos, y la novena se mantiene en cero. El resultado es la paridad del byte,

que es el dato a escribir en el chip de RAM de paridad.

Durante las **lecturas**, la novena entrada es ese mismo bit de paridad, tal como se lee en la RAM. La salida será 0 si ese bit corresponde efectivamente a la paridad de los ocho bits de datos.

En caso de lecturas, las salidas de cada codificador de paridad son combinadas, y si alguna de ellas indica error, es puesta la señal de **error de paridad** PCK.

PCK está controlada, además, por una señal de **habilitación** de verificación de paridad, ENA RAM PCK, programable vía software desde el controlador 8042 (o equivalente). Cuando PCK deviene activa, termina por disparar la interrupción no enmascarable NMI del micro, que normalmente desemboca en un mensaje de error y la parada del sistema.

Refresco

El refresco de RAM parte de la señal de salida del **canal 1 del timer**, programada para generar un pulso cada unos 15 μ s. En condiciones normales (esto es: *que ningún periférico solicite una demora de I/O a través de la línea I/O CH RDY*) se genera con esa cadencia la señal de REFRESH, y se fuerza MEMR, de manera que sobre un banco por vez es forzada una lectura dummy, garantizando la conservación de los contenidos de memoria.



Determinación del ángulo de "skew"

La exploración (scanning) de imágenes -en particular de line arts- exige normalmente el escuadrado (deskewing), que por lo común es interactivo. Sin embargo, es posible encarar de varios modos la determinación automática de la magnitud de este defecto

La interpretación de patrones es, en sí, compleja. Y resulta sumamente difícil encontrar ejemplos que sean simultáneamente comprensibles y no triviales, o excesivamente simplificados.

Si bien los programas de ejemplo suministrados *sí serán excesivamente simplificados* (por la fuerza de que la extensión de esta nota depende en gran medida de la complejidad de ellos), es bueno contar con un ejemplo sencillo y a la vez práctico.

A la vez, puedo asegurarle que la simplificación en los algoritmos es más bien conceptual, y no se traslada a sus resultados, más que aceptablemente buenos.

Bien, terminando ya con el misterio, encontré como ejemplo a

la vez sencillo y realista el **escuadrado** de las imágenes obtenidas mediante un scanner.

Deskew

Se trate de un scanner de mesa o uno de arrastre manual, es casi inevitable que la imagen no se haya explorado "en escuadra", sino que tanto la inclinación de la hoja como el deslizamiento del scanner den como resultado una digitización en la cual las horizontales *no son exactamente horizontales* ni las verticales son tales.

Para cualquiera que sea el destino de la imagen digitizada, se impone la tarea de devolverla a su "escuadra" original, proceso que se conoce como **deskewing**. En general, se asume que la imagen está *sólo ligera-*

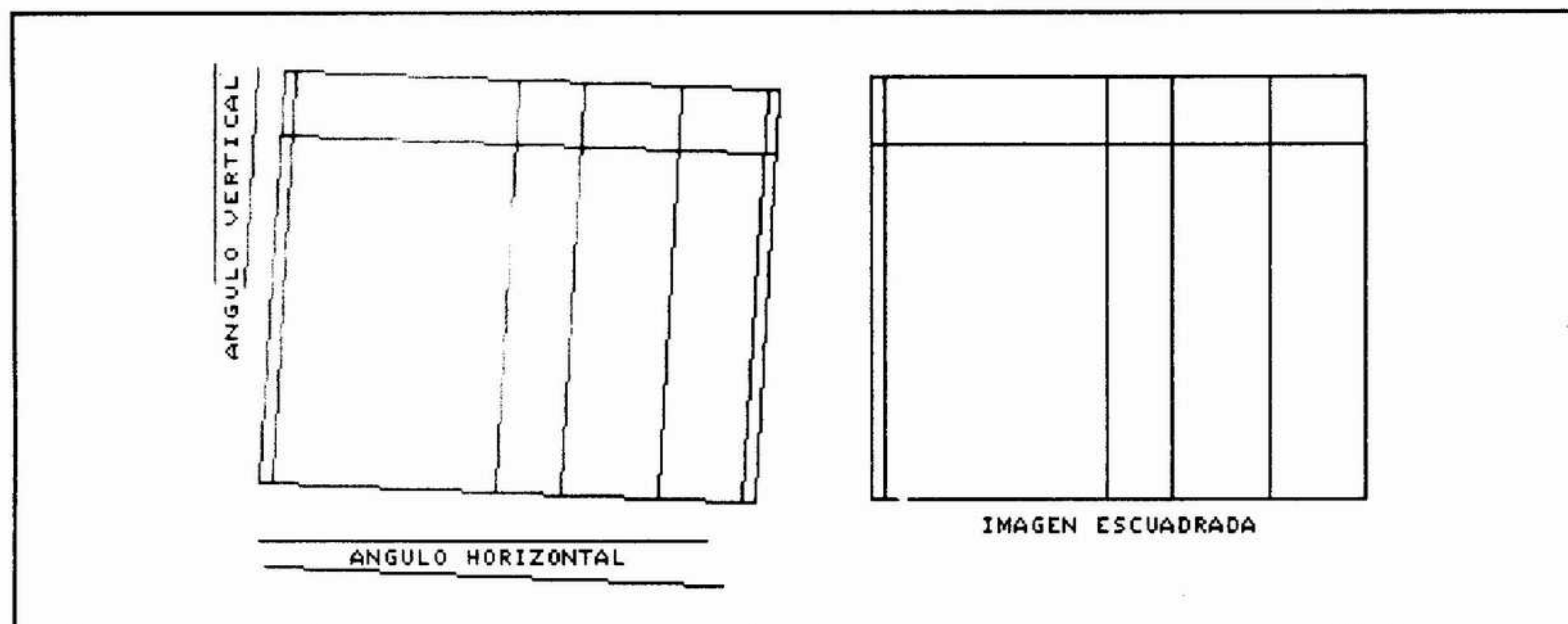


Fig. 1 - Efecto del deskewing

mente desviada de estos ejes, y se corrige esa desviación desplazando zonas. La corrección suele ser independiente para el deslizamiento horizontal y para el vertical.

Como términos análogos que son, no sólo las acciones correctivas son similares; también lo son los diagnósticos de la magnitud del deslizamiento. Por lo tanto, sólo consideraremos el escuadrado según una coordenada (la vertical, es decir, poner horizontales a las líneas que se supone que deberían serlo).

El escuadrado de la imagen se obtiene fácilmente (esto es una manera de decir, porque pueden persistir problemas menos obvios), trasladando píxeles según la regla:

$$P(c,f) \rightarrow P(c,f-a*c)$$

donde $P(c,f)$ representa un píxel en la columna c y fila f , y a es el **factor de deskewing**, que

es la pendiente de las líneas que hubieran debido ser horizontales.

La mayoría de los programas comunes de proceso de imágenes solicitan al usuario la defini-

ción de la dirección de lo que teóricamente debiera haber sido horizontal; luego ejecutan un algoritmo como el descrito. En los ejemplos BASIC sucesivos asumiremos que la instrucción

```

defint a-z
screen 9
x$=input$(1)                                ' Pausa para presentar la imagen
R0=100 :R1=200                                ' Primera y ultima fila a explorar
for P!=-.1 to .1 step .01
  for R=R0 to R1
    Aciertos=0
    for c=0 to 639
      Aciertos=Aciertos-(point(c,R+c*P!)=0)
    next c
    if Aciertos>AciertosMax then
      AciertosMax=Aciertos
      PMax!=P!
    end if
    locate 24,1:print "Ensayando pendiente=";P!;
  next R
next P!
Pte!=PMax!
line (0,100)-(639,100+640*Pte!),12          ' linea de prueba (su
                                              ' pendiente corresponde
                                              ' a la determinada
                                              ' por el programa)

```

Fig. 2 - Determinación de la pendiente por prueba sistemática de modelos


```

defint a-z
screen 9
x$=input$(1)
R0=100 :R1=200
NN=30
DispRC!=0: DispCC!= 1
for B=R0 to R1-NN step NN
  Rm!=0: Cm!=0: Np!=0
  for DR=0 to NN
    for C=0 to 639
      if point(C,B+DR)=0 then
        Rm!=Rm!+DR
        Cm!=Cm!+C
        Np!=Np!+1
      end if
    next C
  next DR
  if Np! then Rm!=Rm!/Np! else Rm!=0
  if Np! then Cm!=Cm!/Np! else Cm!=0
  for DR=0 to NN
    for C=0 to 639
      if point(C,B+DR)=0 then
        DispR!=DR-Rm!
        DispC!=C-Cm!
        DispRC!=DispRC!+DispR!*DispC!
        DispCC!=DispCC!+DispC!*DispC!
      end if
    next C
  next DR
next B
Pte!=DispRC!/DispCC!
line (0,100)-(639,100+640*Pte!),12

```

Pausa para presentar la imagen
Primera y ultima fila a explorar

' linea de prueba (su
' pendiente corresponde
' a la determinada
' por el programa)

Fig. 3 - Determinación de la pendiente en base a la dispersión de puntos

X\$=INPUT\$(1) da oportunidad al usuario para volcar a la pantalla una imagen preformada mediante un capturador como FRIEZE (PaintBrush); de otro modo se deberá reemplazar esta instrucción con el código necesario para formar una imagen.

Hemos adoptado, además, la

convención de que el color de forma sea **negro** (0); el de fondo, que puede ser cualquiera, en general será blanco (15).

Este ejemplo presenta una utilidad adicional: poner en relieve que la inteligencia artificial es muchas veces más artificio que inteligencia. El sofisma frecuente es igualar los términos:

solución de problema complejo=inteligencia artificial

Hoy a nadie le llama la atención que una calculadora evalúe logaritmos, un problema que insumía años a los elaboradores de las famosas tablas de cincuenta o cien años atrás (Lalande, Houel). En 1880, una máquina para calcular logaritmos se hubiera podido hacer pasar por un ejemplo de inteligencia artificial.

Pero, atención: de ninguna manera pretendo devaluar los logros de las técnicas que auténticamente se autodenominan de AI, que simplemente están a la vista. Sí quiero dejar sentado que, en primer lugar, no hay magia, y en segundo, que la inteligencia sigue siendo la del programador y la del operador. En el programa, no existe.

Criterios

A veces es más difícil plantear un problema que resolverlo. Resolverlo no deja de ser un problema técnico, que exige un cierto dominio (a veces mucho) de lógica y matemática, y la conciliación de requerimientos y disponibilidades (memoria, tiempo, etc.).

Plantearlo, en cambio, requiere alimentar esa máquina. Poner en términos lógicos y matemáticos sencillos la realidad. Esto se denomina **modelización**: lo que se propone no es -en general- el *reflejo fiel* de la realidad, sino un modelo que reproduzca los aspectos relevantes, que

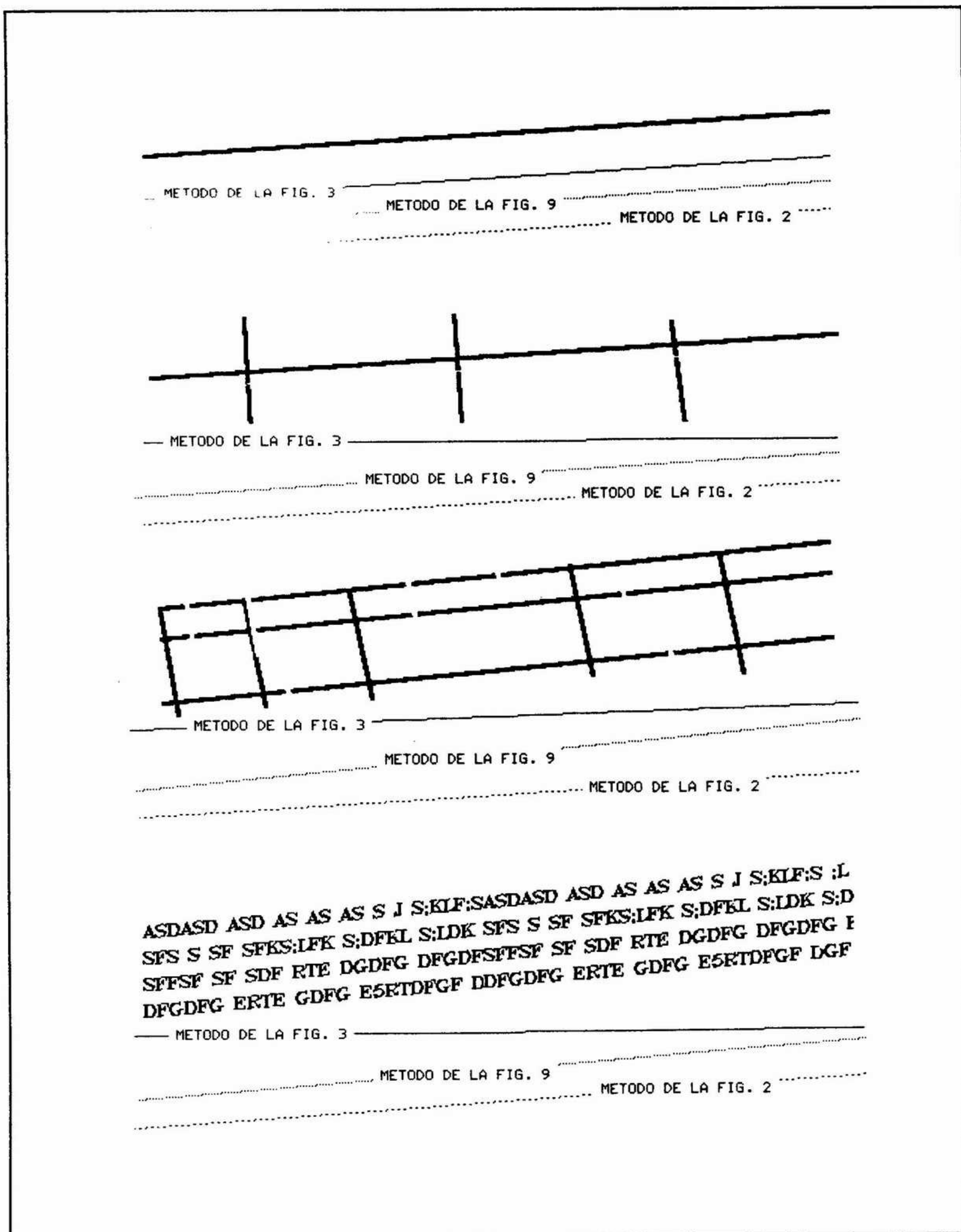


Fig. 4 - Resultados de aplicar los diferentes métodos a varios casos de prueba


```

defint a-z
screen 9
x$=input$(1)
R0=100 :R1=200
for R=R0 to R1
  for C=0 to 639
    IF point(C,R)=0 THEN
      PSET(c-1,r),0 : PSET(c-2,r),0
      PSET(c,r-1),0 : PSET(c-1,r-1),0 : PSET(c-2,r-1),0
      PSET(c,r-2),0 : PSET(c-1,r-2),0 : PSET(c-2,r-2),0
    END IF
  next C
next R

for R=R0 to R1
  for C=0 to 639
    if point(C,R)>point(C,R+1) then pset(C,R),0 else pset(C,R),15
  next C
next R
for R=R0 to R1
  for C=0 to 639
    x= -(point(C,R)=0)-(point(C+1,R)=0)-(point(C+2,R)=0)-
      -(point(C,R+1)=0)-(point(C+1,R+1)=0)-(point(C+2,R+1)=0)
    if x<3 then pset(C,R),15
  next C
next R

```

· Pausa para presentar la imagen
· Primera y ultima fila a explorar

permita un manejo matemático suficientemente sencillo y que pueda parametrizarse con términos efectivamente medibles.

En cuestiones físicas, la dificultad reposa en que ciertos parámetros no son fácilmente medibles, y en que muchas veces es muy difícil describir las leyes o la geometría de un sistema.

En problemas como el que nos ocupa, y que suelen ser el objetivo de técnicas de AI, la dificultad reside en la siguiente paradoja: Aspectos del problema que son evidentes, incluso para personas muy poco dotadas, son muy difíciles de describir en términos lógicos elementales. Por ejemplo, la diferencia entre las formas de una I y una J es difícil de describir (*salvo en términos informales*), y muy fácil de identificar.

Fig. 5 - Preprocesamiento: detección de bordes y supresión de patrones verticales

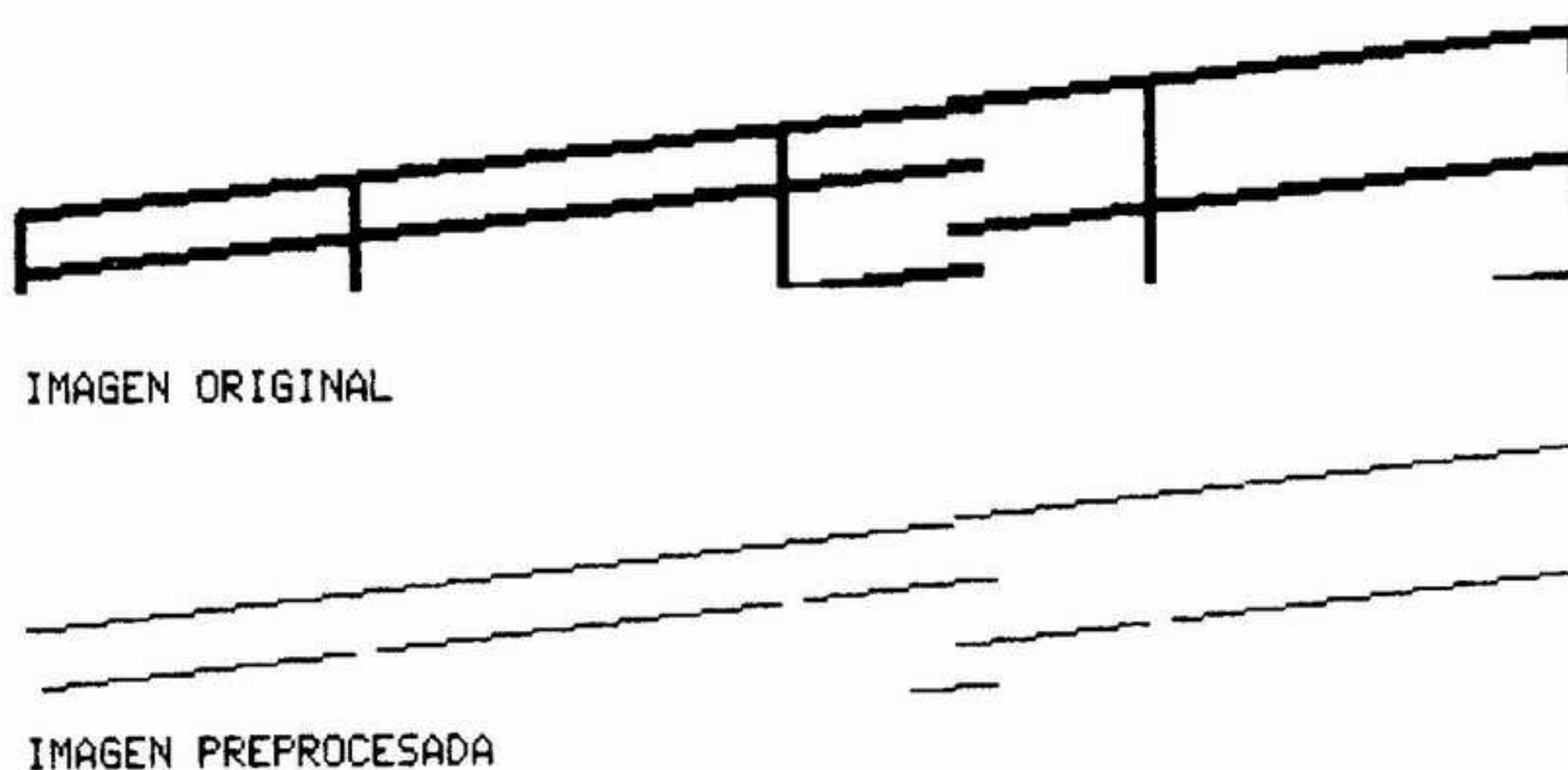


Fig. 6 - Efecto del preprocesamiento sobre un caso típico

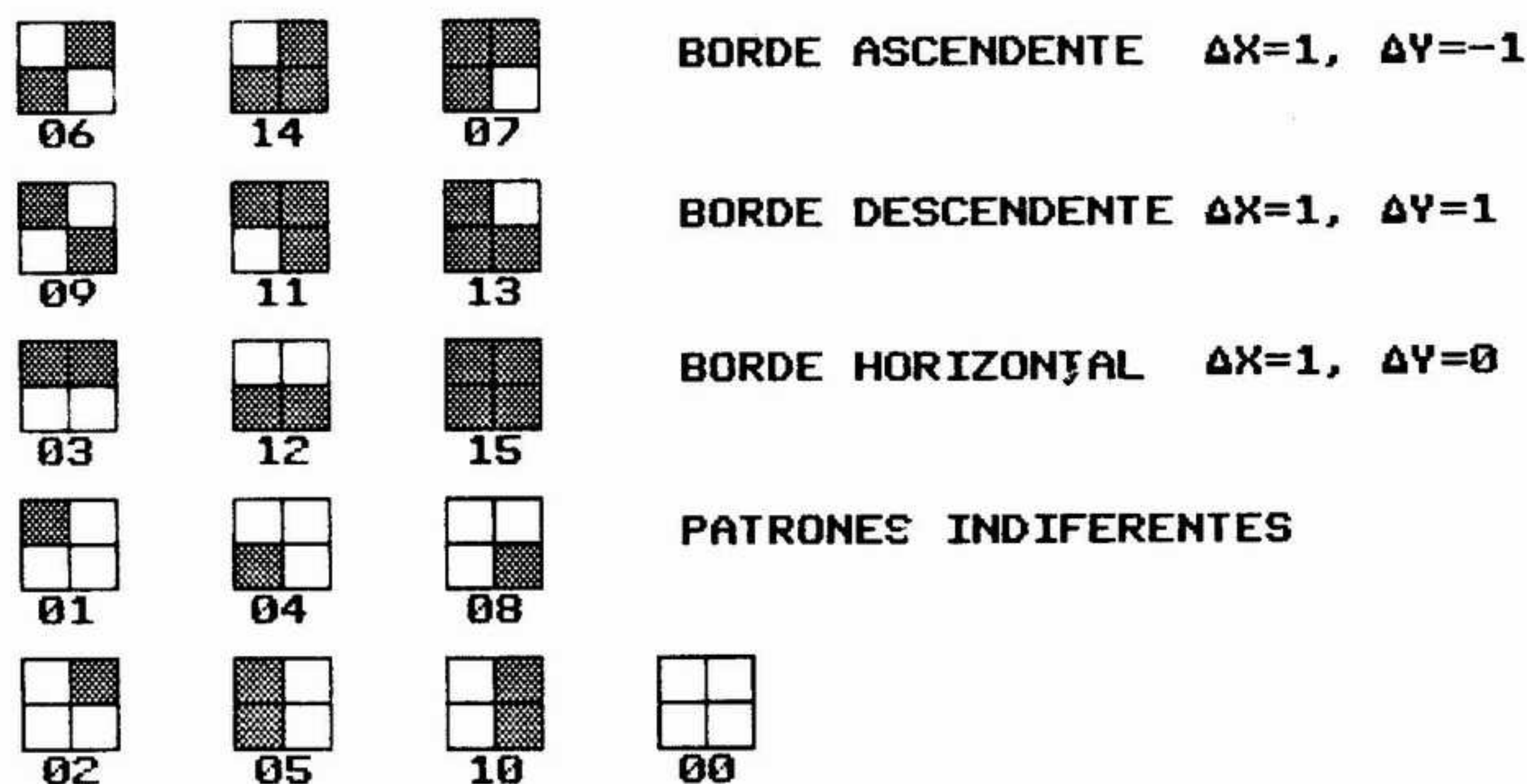


Fig. 7 - Exploración mediante un operador local de cuatro píxeles

Identificación de imágenes

Esto nos conduce a algunas aproximaciones en la identificación de imágenes. La primera de ellas adopta el enfoque clásico en informática: emplear la computadora como generador de variantes (*como ordenador*, cumpliendo literalmente la nomenclatura francesa a la que adhieren los españoles). La segunda, es profundizar el examen y tratar de extraer patrones.

Por último, ya dentro de la órbita de AI, se intenta emular nuestro mecanismo de conocimiento, o, mejor dicho, de percepción. Recalco: lo que se trata de reproducir no es en realidad inteligencia, sino etapas que son previas a ella, como el reconocimiento de formas.

No hay nada inteligente en la manera en como reconocemos

una letra A, la inclinación dominante en una figura, o una silueta humana. Tampoco podemos explicar como lo hacemos.

A falta de explicaciones biológicas suficientes, se intentan aproximaciones experimentales, a las que volveremos luego.

La Aproximación Top-Down

Un posible enfoque para el reconocimiento de un objeto es la **comparación directa** contra un modelo completo de la imagen. En general, se requerirán diferentes modelos, cada uno con distintas posiciones y orientaciones.

Si se me permite una sobresimplificación, este método equivale a superponer, uno a uno, una familia de modelos con la imagen a identificar, buscando aquel que produce la mejor coincidencia. En la práctica, es-

to implica determinar un **objetivo de coincidencia** (por ejemplo, cantidad de píxeles superpuestos/cantidad de píxeles totales), y parametrizar las variantes de un mismo modelo.

Como el número de variantes puede ser muy elevado, es importante reducir en lo posible el número de ensayos.

El caso que adoptamos como ejemplo es muy sencillo: una recta de inclinación desconocida, próxima a la horizontal.

Primero vamos a ensayar la aproximación más grosera, que consiste en proponer una familia de rectas y asignar a cada una un puntaje, resultante de la cantidad de píxeles que se superponen entre la imagen a analizar y la recta intentada. El programa de la figura 2 muestra este ensayo, y es considerablemente lento.


```

defint a-z
screen 9
x$=input$(1)
R0=40 :R1=100

for R=R0 to R1
  for C=0 to 639
    IF point(C,R)=0 THEN
      PSET(c-1,r),0 PSET(c-2,r),0
      PSET(c,r-1),0: PSET(c-1,r-1),0: PSET(c-2,r-1),0
      PSET(c,r-2),0: PSET(c-1,r-2),0: PSET(c-2,r-2),0
    END IF
  next C
next R

for R=R0 to R1
  for C=0 to 639
    if point(C,R)>point(C,R+1) then pset(C,R),0 else pset(C,R),15
  next C
next R

for R=R0 to R1
  for C=0 to 639
    X=-(point(C-2,R)=0)-(point(C-1,R)=0)-(point(C,R)=0)-
      -(point(C+2,R)=0)-(point(C+1,R)=0)
    IF X<3 THEN PSET(c,r),15
  next C
next R

for R=R0 to R1
  for C=0 to 639
    SI=-(point(C,R)=0)
    SD=-(point(C+1,R)=0)
    II=-(point(C,R+1)=0)
    ID=-(point(C+1,R+1)=0)
    IF SI=0 AND SD=1 AND II=1 AND ID=0 THEN ASCE=ASCE+1
    IF SI=1 AND SD=0 AND II=0 AND ID=1 THEN DESC=DESC+1
    IF SI=1 AND SD=1 AND II=0 AND ID=0 THEN PLN=PLN+1
  next C
next R
PEND!=(ASCE-DESC)/(PLN+ASCE+DESC)
LINE (0,R1)-(639,R1-PEND!*640),12

```

Fig. 8 - Determinación de la pendiente por análisis local de pixels

Los resultados obtenidos son buenos, en tanto que la imagen sea efectivamente una recta; imágenes más complejas (p.

ej., simplemente dos rectas) llevan a resultados erróneos.

Una simplificación

En el algoritmo mencionado, el tiempo se consume en numerosos ciclos de prueba, que quizás se podrían abreviar. Ciertas indicaciones globales nos pueden decir de antemano que la imagen analizada y el modelo propuesto no se llevan bien, *sin necesidad de analizar punto a punto*. Incluso, es posible en base a ese tipo de información limitar bastante la cantidad de modelos a probar.

Si la imagen y el modelo presentan una coincidencia razonable, es de esperar que ciertos parámetros tales como el "**centro de gravedad**" (posición), y la medida de la **dispersión** horizontal y vertical (ancho y largo) sean similares.

Esto significa que sólo deberemos probar con modelos cuyo centro de gravedad y dispersión se asemejen a los de la imagen, que se calculan al principio y una sola vez. Las coordenadas del centro de gravedad son, simplemente, los **valores medios** de las filas y columnas de los puntos, cuyo número total es N:

$$N = \sum P(c,f)$$

$$G_f = (\sum P(c,f) * f) / N$$

$$G_c = (\sum P(c,f) * c) / N$$

$P(c,f)$ vale 1 si en la posición c,f hay un punto y cero en caso contrario (en nuestros ejem-

plos, basados en figuras negras sobre fondo blanco. $P(c,f)$ vale 1 si $\text{point}(c,f)=0$, y 0 en caso contrario).

Las dispersiones se definen como:

$$D_f = \sqrt{(\sum P(c,f) \cdot (f-G_f)^2)/N}$$

$$D_c = \sqrt{(\sum P(c,f) \cdot (c-G_c)^2)/N}$$

y se pueden calcular, mejor, mediante las relaciones

$$D_f = \sqrt{(\sum P(c,f) \cdot f^2 - N G_f^2)/N}$$

$$D_c = \sqrt{(\sum P(c,f) \cdot c^2 - N G_c^2)/N}$$

que se derivan de las anteriores, y permiten evaluar todos los parámetros en un solo ciclo.

Podríamos tratar de aplicar estas restricciones al caso de una recta, pero con un objeto tan sencillo *podemos ir mucho más lejos*: como lo único que interesa es determinar la pendiente, se puede recurrir al *-famoso-* método de cuadrados mínimos, que busca **minimizar la dispersión** entre los puntos de la recta a analizar y los de la propuesta.

Para no entrar demasiado en detalles, digamos que la pendiente de la recta que mejor se ajusta según este criterio viene dada por:

$$a = S_{xy}/S_{xx}$$

donde S_{xy} es la **dispersión "cruzada"**:

$$S_{yx} = \sum P(c,f) (f-G_f)(c-G_c)$$

y S_{xx} es el cuadrado de la dispersión horizontal:

$$S_{xx} = \sum P(c,f) (c-G_c)(c-G_c)$$

```
defint a-z
screen 9
x$=input$(1)
R0=100 :R1=200

' Pausa para presentar la imagen
' Primera y ultima fila a explorar

for R=R0 to R1
  for C=0 to 639
    IF point(C,R)=0 THEN
      PSET(c-1,r),0: PSET(c-2,r),0
      PSET(c,r-1),0: PSET(c-1,r-1),0: PSET(c-2,r-1),0
      PSET(c,r-2),0: PSET(c-1,r-2),0: PSET(c-2,r-2),0
    END IF
  next C
next R

for R=R0 to R1
  for C=0 to 639
    if point(C,R)>point(C,R+1) then
      pset(C,R),0
    else
      pset(C,R),15
    end if
  next C
next R

for R=R0 to R1
  for C=0 to 639
    BYTE=0
    IF (point(C,R) =0) THEN BYTE=BYTE OR 1
    IF (point(C+1,R) =0) THEN BYTE=BYTE OR 2
    IF (point(C,R+1) =0) THEN BYTE=BYTE OR 4
    IF (point(C+1,R+1)=0) THEN BYTE=BYTE OR 8
    IF BYTE=&h06 OR BYTE=&h07 OR BYTE=&h0E THEN _
      ASCE=ASCE+1
    IF BYTE=&h09 OR BYTE=&h0B OR BYTE=&h0D THEN _
      DESC=DESC+1
    IF BYTE=&h03 OR BYTE=&h0C OR BYTE=&h0F THEN _
      PLN=PLN+1
  next C
next R
PEND!=(ASCE-DESC)/(PLN/2+ASCE+DESC)
LINE (0,R1)-(639,R1-PEND!*640),0
' linea de prueba (su
' pendiente corresponde
' a la determinada
' por el programa)
```

Fig. 9 - En esta versión se emplea manifiestamente el operador local

Objetivo:

```
for R=R0 to R1
  for C=0 to 639
    IF point(C,R)=0 THEN
      PSET(c-1,r),0 PSET(c-2,r),0
      PSET(c,r-1),0: PSET(c-1,r-1),0: PSET(c-2,r-1),0
      PSET(c,r-2),0: PSET(c-1,r-2),0: PSET(c-2,r-2),0
    END IF
  next C
next R
for R=R0 to R1
  for C=0 to 639
    if point(C,R)>point(C,R+1) then
      pset(C,R),0
    else
      pset(C,R),15
    end if
  next C
next R
for R=R0+1 to R1-1
  for C=C0+1 to C1-1
    BYTE=0
    IF (point(C,R) =0) THEN BYTE=BYTE OR 1
    IF (point(C+1,R) =0) THEN BYTE=BYTE OR 2
    IF (point(C,R+1) =0) THEN BYTE=BYTE OR 4
    IF (point(C+1,R+1)=0) THEN BYTE=BYTE OR 8
    IF BYTE=&h06 OR BYTE=&h07 OR BYTE=&h0E THEN_
      PATRON!(NP,1)=PATRON!(NP,1)+1
    IF BYTE=&h09 OR BYTE=&h0B OR BYTE=&h0D THEN_
      PATRON!(NP,2)=PATRON!(NP,2)+1
    IF BYTE=&h03 OR BYTE=&h0C OR BYTE=&h0F THEN_
      PATRON!(NP,3)=PATRON!(NP,3)+1
  next C
next R
TOTAL!=0
FOR I=1 TO 3
  TOTAL!=TOTAL!+PATRON!(NP,I)
NEXT I
FOR I=1 TO 3
  PATRON!(NP,I)=PATRON!(NP,I)/TOTAL!
NEXT I
RETURN
```

Fig. 10 - Esta subrutina preprocesa la imagen y calcula la función objetivo para un caso dado

Evaluando estos términos se reconoce la pendiente sin nece-

sidad de "probar" diferentes (y teóricamente infinitas) rectas.

En casos más complejos, evaluaciones similares sólo permitirán acotar algunos de los parámetros que determinan las variaciones de los modelos a ensayar, pero con ello reduciendo enormemente el número de ciclos de prueba.

Fallas

El problema de este método es que se espera una correspondencia muy ajustada entre el modelo y la imagen, la cual falla irremediablemente si la imagen incluye elementos adicionales o se aparta de lo esperado; menor aún es la habilidad para reconocer patrones más elásticos.

En la figura 4b se muestra una imagen de un conjunto de rectas a escuadrar. Si bien la dirección es uniforme y obvia, es difícil que un método como el anterior pueda determinarla.

Reconocimiento de patrones

Un enfoque diferente viene dado por el reconocimiento de **patrones locales**. En general, se trata de reconocer configuraciones de puntos próximos, con *prescindencia de la globalidad de la imagen*; luego se deberá integrar el resultado de este reconocimiento.

El caso del "deskew" es sencillo, porque no hace falta más que el reconocimiento local. Manteniendo la idea de esquemas sencillos pero demostrativos, la exploración se hará en **grupos de cuatro celdas**. Se

define un **operador movil**, que devuelve un resultado a partir del examen de un grupo de celdas vecinas.

En nuestro caso, partimos de la observación de patrones típicos de líneas. De las 16 combinaciones posibles para cuatro celdas, observamos que las números 0, 1, 2, 4, 5, 8 y 10 son indiferentes; las números 3 y 12 corresponden a los bordes superior e inferior de líneas horizontales. Cuando una línea está ligeramente inclinada hacia arriba, aparecerá alguno de los patrones 6, 7, o 15; corresponderán los números 9, 11, y 14 a las inclinadas hacia abajo (figura 7).

Podemos ir un poco más lejos: Llamemos **P** (plano) a la suma de los casos 3 y 12, **A** (ascendente) a la de los números 6, 7 y 15 y **D** (descendente) a la suma de 9, 11 y 14. Si nos aseguramos de que el ancho de las líneas sea de un píxel, se cumplirá:

$$\text{pendiente} = (A - D) / (A + D + P/2)$$

Veamos porqué:

Los casos **P** se habrán computado **por duplicado**, porque si aparece el patrón 3 en una posición, aparecerá el 12 al explorar la fila siguiente. Dividido **P** por 2, se tiene el ancho de los tramos horizontales.

Agregando **D** y **A**, se totaliza el ancho de la recta. Cada patrón de tipo **A** implica un desplazamiento de **un píxel hacia arriba**, y cada uno de tipo **D**, de

```
defint a-z
screen 9
DIM Patron!(200,3),Pend!(200)
x$=input$(1)

OPEN"i",#1,"PATRONES"
NP=0
WHILE NOT EOF(1)
  NP=NP+1
  input#1,Pend!(NP)
  FOR I=1 TO 3: input#1,Patron!(NP,I): NEXT I
WEND
CLOSE#1

NP=NP+1
gosub Objetivo
PRINT "Valor de la pendiente:"; input Pend!(NP)

OPEN"o",#1,"PATRONES"
FOR IP=1 TO 21
  PRINT#1,Pend!(IP)
  FOR I=1 TO 3
    PRINT#1,Patron!(IP,I)
  NEXT I
NEXT IP
CLOSE#1
```

Fig. 11 - Programa de "aprendizaje"

uno hacia abajo; la diferencia nos da la excursión vertical de la línea.

Por último, el cociente entre la excursión vertical y el ancho (horizontal) determina la pendiente.

Preprocesamiento

Como se dijo, lo anterior es válido si el ancho de la línea es de un solo píxel. Este requerimiento nos lleva a un proceso previo de la imagen, para facilitar su interpretación. En general, en este tipo de problemas (y cual-

quiera sea la metodología empleada) es necesario siempre un preproceso para mejorar la definición, eliminar "ruido" y acentuar las características de interés.

En nuestro caso, elegimos una determinación de **bordes horizontales**. Si bien esto no pasa de un artificio lógico, que se puede integrar al cálculo anterior, preferimos efectuar esta transformación directamente sobre la imagen, de manera que sus efectos sean fácilmente apreciables visualmente.


```

defint a-z
screen 9
DIM Patron!(200,3),Pend!(200), Coinc!(200), MejorModelo(200)
x$=input$(1)

OPEN"1",#1,"PATRONES"
NP=0
WHILE NOT EOF(1)
  NP=NP+1
  input#1,Pend!(NP)
  FOR l=1 TO 3: input#1,Patron!(NP,l): NEXT l
WEND
CLOSE#1
NP=NP+1

R0=100: R1=200
gosub XXXXX

FOR IP=1 TO NP-1
  MejorModelo(IP)=IP
  Coinc!(IP)=0
  FOR l=1 TO 3
    Coinc!(IP)=Coinc!(IP)+(Patron!(NP,l)-Patron!(IP,l))^2
  NEXT l
NEXT IP
FLAG=-1
WHILE FLAG
  FLAG=0
  FOR IP=1 TO NP-2
    IF Coinc!(IP)>Coinc!(IP+1) THEN
      SWAP Coinc!(IP),Coinc!(IP+1)
      SWAP MejorModelo(IP),MejorModelo(IP+1)
      FLAG=-1
    END IF
  NEXT IP
WEND
LOCATE 1,1
PRINT MejorModelo(1),Coinc!(1),Pend!(MejorModelo(1)),
PRINT MejorModelo(2),Coinc!(2),Pend!(MejorModelo(2)),

```

Fig. 12 - Programa de diagnóstico

El programa de la figura 5 compara cada píxel con el inferior, y deja un punto sólo cuando exis-

te un cambio de blanco a negro (borde superior de las formas). Si se corre este breve programa

sobre una imagen compleja, se observará que además de reducir las líneas a un píxel de ancho, *elimina en gran medida los componentes verticales de la imagen*, acentuando las tendencias horizontales (este procedimiento equivale a un filtro pasaaltos para la coordenada vertical).

Un algoritmo de determinación de dirección

Combinando este preproceso con el algoritmo discutido más arriba, se obtiene el ejemplificado en la figura 8. Mediante éste, es posible determinar la inclinación dominante en figuras complejas, siempre que estén constituidas mayoritariamente por rectas casi horizontales o casi verticales.

Aprendizaje

Los enfoques analizados, aún bien diferentes, comparten una característica: toda la lógica, simple o compleja, está **prevista y fijada** en el algoritmo mismo.

En un caso sencillo como el tomado de ejemplo, esto es ventajoso, porque simplifica el algoritmo y le otorga un buen grado de eficiencia.

Pero si las características a identificar o evaluar en la imagen fueran menos concretas, la elaboración de un algoritmo más elástico podría tornarse sumamente complicada. Una de las causas es la dificultad en

establecer pautas que caractericen adecuadamente modelos complejos.

Uno de los enfoques habitualmente adoptados en AI (*inteligencia artificial*) se basa, precisamente, en definir concretamente el **método de exploración**, dejando más o menos libre la interpretación de los patrones obtenidos.

Esto significa que se harán pocas hipótesis sobre cómo interpretar los patrones, quizás las mínimas para facilitar el aprendizaje, que es la etapa siguiente.

Un algoritmo de este tipo puede, entonces, obtener de una

imagen una caracterización numérica (booleana, multivaluada o basada en pesos reales), sobre la cual *no se realizan mayormente hipótesis de interpretación*. La etapa de aprendizaje consiste en suministrar al programa diferentes ejemplos, junto con la caracterización correcta.

El programa asocia estas caracterizaciones con los patrones obtenidos, y, posteriormente, frente a una incógnita, intentará identificarla con el o los modelos más parecidos.

La elaboración de algoritmos de este tipo es compleja, por un lado debido al problema de re-

conocimiento del modelo, y por el otro porque exige el manejo de grandes cantidades de memoria. La etapa de aprendizaje no es trivial: es decididamente prolongada, y se debe cuidar de que se hayan suministrado ejemplos suficientes y no desviados por alguna tendencia.

Deseando que el lector pueda hacerse una idea más concreta (*más práctica*) de este mecanismo, vamos a retomar el problema anterior, con la intención de construir un algoritmo de aprendizaje, para luego retomar esta discusión sobre AI.

Tomar contacto directo con dificultades técnicas, lados flacos,

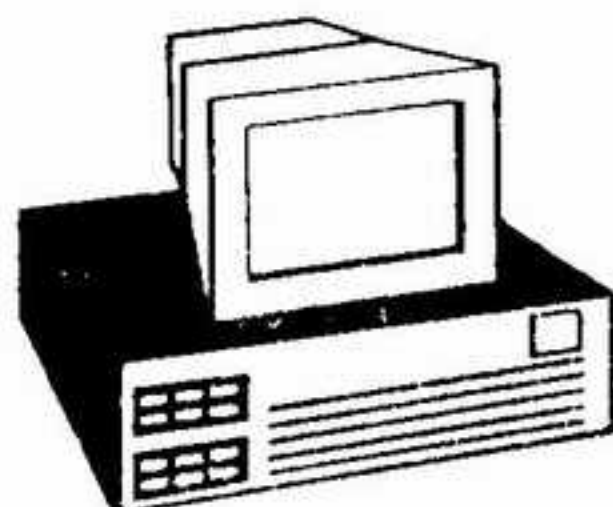
LABORATORIO DIGITAL

SERVICIO TECNICO ESPECIALIZADO

CANJE DE FUENTES EN EL ACTO

REPARACIONES DE:

- RIGIDOS, MONITORES COLOR CGA, VGA, SVGA y de TV COLOR
- TAPE BACKUPS, FLOPPYS, IMPRESORAS TODAS LAS MARCAS
- LINEA COMMODORE EN GENERAL. AMIGA 500 - 600 - 2000 - 3000 y 4000
- LINEA DE NOTEBOOK Y LAPTOP, FAX, AUDIO y VIDEO



PRESUPUESTOS SIN CARGO. ATENCION AL GREMO. TRABAJOS PARA EL INTERIOR DEL PAIS. GARANTIA ESCRITA Y REAL PARA TODO TRABAJO

**SERVICIO DE MANTENIMIENTO MENSUALIZADO
PARA COLEGIOS, EMPRESAS E INSTITUTOS**

Horario
9.30 a 18.30 hs.

Av. de Mayo 822 - 4º piso of. 5 - CP 1084 Capital - Teléfono 342-1291/0646

palpar las posibles mejoras, es más que importante en muchos casos. En éste, diría que es imprescindible, porque se trata de un ámbito comparable con el de la teoría de la Relatividad: comprenderla exige leer con lápiz, papel, y voluntad dispuestos; de otro modo, puede Ud. leerla en *Reader's Digest*, y obtendrá sólo la ilusión de que entendió algo, y el consuelo de que podrá hablar de ello en alguna reunión social.

Un modelo de aprendizaje

Cabe advertir desde el principio que el siguiente algoritmo incluye una "trampa": *me he tomado la libertad de incluir algunas hipótesis de las que se debiera prescindir en un enfoque más elástico*, para mantener el resultado simple y practicable.

Volviendo a los patrones analizados más arriba, y a su clasificación en cuatro grupos (uno de los cuales es irrelevante), nos basaremos en la hipótesis *de que dicha clasificación es conducente*.

Pero no agregaremos hipótesis alguna acerca de la relación entre la pendiente de las líneas y los acumulados **P**, **A** y **D**: las características de esa relación surgirán de un aprendizaje, en el que suministraremos ejemplos, junto con los valores de pendientes que nos parezcan correctos.

Una primera aproximación es simplemente coleccionar todos los casos dados como ejemplos

en forma de vectores conteniendo **P**, **A**, **D** y el valor de pendiente suministrado. Para evaluar la correspondencia entre un patrón **P-A-D** y alguno de los almacenados emplearemos un criterio de **dispersión mínima**. Los valores de **P**, **A** y **D** se **normalizan** dividiéndolos por el total **P+A+D**; la dispersión entre dos patrones (el incógnito y uno cualquiera de los almacenados) es

$$(P-P')^2 + (A-A')^2 + (D-D')^2$$

y se adoptará como más parecido el modelo para el cual ella sea mínima.

El programa de la figura 11 ejemplifica esta idea: ante cada imagen suministrada, el programa intenta aparearla contra alguno de los patrones ya conocidos, determinando un valor probable. Si además se le suministra el valor correcto, el nuevo caso será incorporado a la memoria.

Esta forma de reconocimiento es bastante grosera, porque no ofrece formas de sintetizar los patrones incorporados, que se van acumulando sin límite, y determina la semejanza en base a simple dispersión de parámetros. Si los parámetros de cada patrón fuesen más numerosos (por ejemplo, si renunciáramos a nuestra hipótesis que permite clasificar los grupos), se vería que hay parámetros irrelevantes, otros decisivos, y algunos cuya influencia se da en forma conjunta con otros.

La flexibilidad de un algoritmo como este depende de su capacidad para determinar estas situaciones en base a la experiencia adquirida, lo que no es fácil de conseguir.

Para no complicar nuestro ejemplo, hasta ahora sencillo, solamente consideraremos cómo evitar la acumulación indefinida de patrones.

La nueva hipótesis es que dos ejemplos con igual valor (en este caso, de la pendiente) pueden dar patrones muy similares, o bien patrones bastante diferentes. En el primer caso, no tiene sentido almacenar cada instancia por separado, y se puede mantener un promedio de los casos parecidos.

El criterio sigue siendo el mismo: si la dispersión es pequeña, se asume que el nuevo ejemplo *es un caso más de lo que ya se había registrado*; se lo incorpora al promedio (esto obliga a conservar, para cada patrón, la cantidad de ejemplos en que ha aparecido).

Si la dispersión es grande, se lo registra como un caso esencialmente diferente.

Volviendo a la AI

Una de las conclusiones que se pueden extraer es que cuando se renuncia a hipótesis de base, se termina por manejar conjuntos de patrones cuya conexión con los resultados **deja de ser evidente**. Sólo el aprendizaje prolongado y un reconoci-

miento complejo pueden extraer información de un nuevo patrón suministrado, *pero ya nadie sabe cómo ocurre esto*.

Desde luego, *tampoco el propio programa lo sabe*: la AI es artificial, y hasta el presente, bastante mecánica.

El ejemplo presentado es una aproximación que en sí **no constituye un ejemplo de AI**, sino un ensayo para experimentar algunos de sus aspectos; el enfoque presentado tampoco es el único posible en las aproximaciones que se autodenominan de AI.

Pero así y todo, el lector podrá apreciar que la denominación AI es excesivamente pretenciosa, porque intenta transformar una capacidad de aprendizaje limitada en inteligencia.

Como el rótulo de AI es deseable, es posible que el lector encuentre software que (*ya decididamente*), no pertenece a esta categoría, y con el que se acude a la gastada estrategia de hacer creer que AI **equivale a muchos IFs**: se trata de algoritmos extensos, con capacidad de manejar muchas variantes y con resultados que tal vez sorprendan, pero la lógica ha sido *envasada en origen*.

La realidad es que, si bien métodos como el ejemplificado son positivamente fructíferos, poco tienen de inteligentes. El programa *no puede desbordar nunca las hipótesis de base* (aunque sean pocas). Si bien

puede establecer identificaciones positivas de objetos complejos, jamás podrá explicarlas en términos de causalidad.

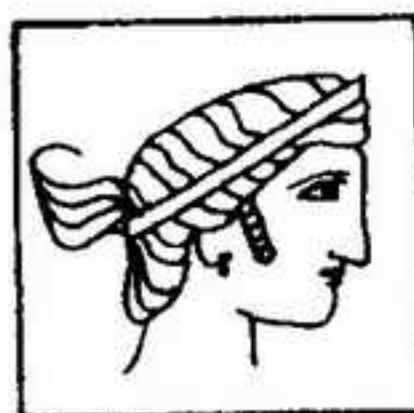
El test de Turing

Como una definición operativa de inteligencia, **Alan Turing** propuso que si es posible interrogar a un sistema, sin saber si las respuestas provienen de un tal sistema o de un ser humano, y sin poder discernir (estadísticamente) esa procedencia, el sistema debe considerarse inteligente.

El problema es que -por fuerza-

las aplicaciones prácticas del test de Turing son adrede restringidas a un cierto ámbito. Se reclama en numerosos casos haber cumplido con el test, pero entre las restricciones de base de conocimiento se suelen mezclar restricciones de adaptación a problemas nuevos.

Un ensayo no restricto implicaría no sólo una base de conocimiento abierta, sino matices más sutiles, como la capacidad de saltar por sobre el objetivo aparente del problema para abordar **el real**.



Palhas S.R.L.



HEWLETT PACKARD

**DISTRIBUIDOR Y CENTRO
AUTORIZADO DE SERVICIO**

REPARACIÓN DE PC'S Y PERIFÉRICOS DE
HEWLETT-PACKARD EN GARANTÍA Y FUERA DE
GARANTÍA.

TOMAMOS SU VIEJA IMPRESORA DE
HEWLETT-PACKARD COMO PARTE DE PAGO
PARA LA COMPRA DE UNA NUEVA.

PERÚ 764 - 2º PISO - CAPITAL FEDERAL
TEL./FAX: 362-1300/9387



Experimentando con su PC

Medición acústica de distancias

La propagación del sonido puede servir de base para la determinación de distancias. Una de sus aplicaciones es la construcción de dispositivos de apuntamiento no convencionales



El hecho de que el sonido se propaga con una velocidad fija permite determinar distancias entre la fuente de sonido y un receptor. Es posible construir dispositivos que emitan sonido de frecuencias apropiadas y señalen el momento de su arribo al receptor. Entonces, basta medir la demora desde una PC para establecer la distancia aproximada. Combinando elementos de este estilo se pueden implementar dispositivos de apuntamiento bastante curiosos.

Un poco de física

La propagación del sonido es un fenómeno ondulatorio. Una onda de compresión-expansión se propaga a través de un fluido (que generalmente es aire) con una velocidad *que depende sólo de las características del fluido, y no de las condiciones de emisión.*

Para el aire, esta velocidad depende de la presión, temperatu-

ra y humedad, pero en condiciones normales es de aproximadamente 330m/s, o, más apropiadamente para la escala de tiempos que vamos a manejar, de 0.33 mm/ μ s.

La longitud de onda λ resulta de dividir velocidad de propagación por frecuencia. Por ejemplo, para 1 KHz (10^3 s^{-1}), resulta de 33 cm.

El efecto de un obstáculo en el camino de la onda sonora depende de la relación entre el tamaño del obstáculo y la longitud de onda empleada. Cuando la dimensión característica del obstáculo es pequeña en comparación con la longitud de onda (típicamente, menor que $1/4 \lambda$), el efecto es mínimo (la onda "rodea" el obstáculo, o, más precisamente, se difracta a su alrededor). Esto indica que si se desea determinar la presencia de un objeto por reflexión, o por interposición en medio de un haz sonoro, deberemos em-

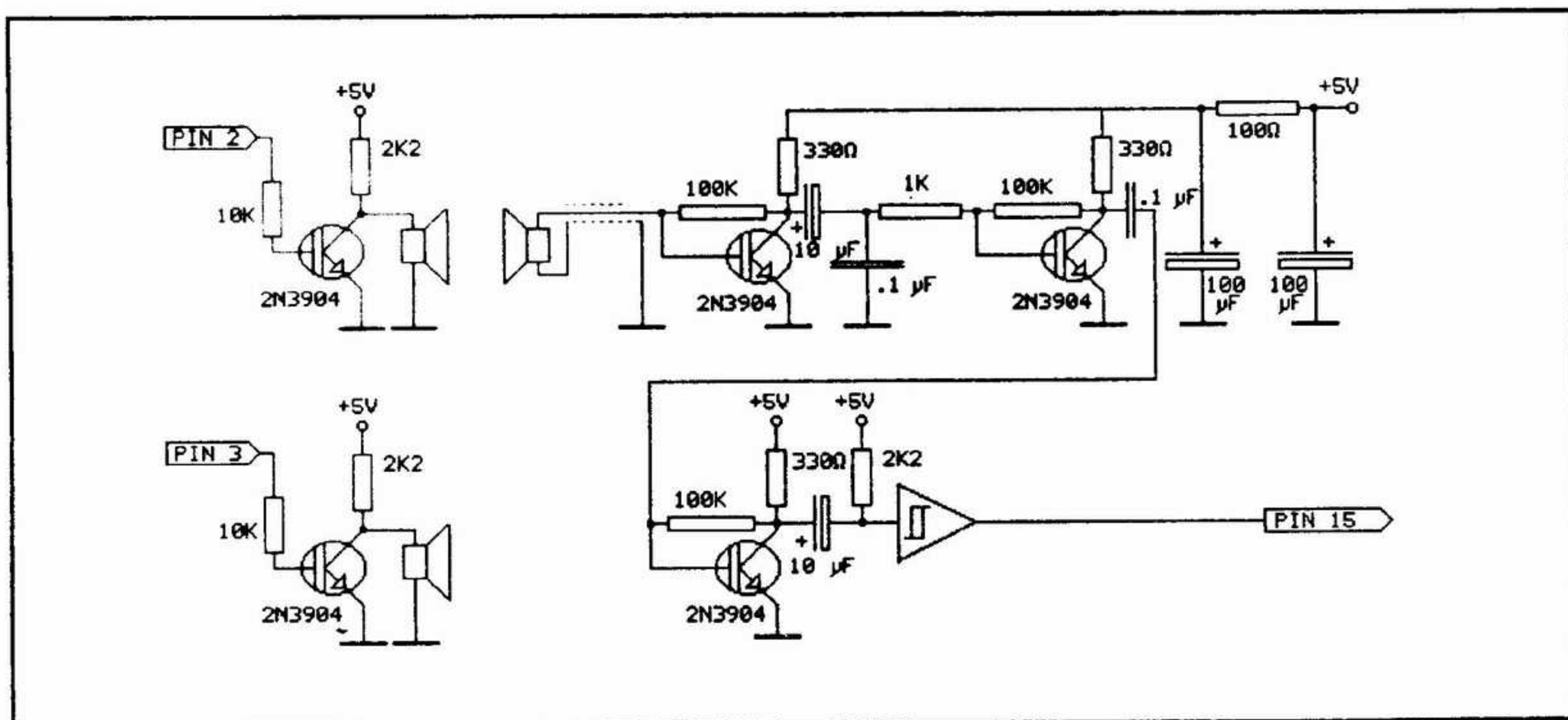


Fig. 1 - Diagrama esquemático del emisor y del sensor

plear longitudes de onda **suficientemente cortas** (o sea, frecuencias suficientemente altas). La misma limitación aparece si se desea sensor por reflexión: para que un área de unos pocos cm de ancho sea suficiente, será necesario que la longitud de onda sea de 1 cm o menos; la frecuencia, por lo tanto, debe superar los 33 KHz.

Además, se debe minimizar la influencia del sonido ambiente. Por último, se debe tomar en cuenta un detalle que la experiencia normal en audio no considera: el comportamiento transiente del detector. Cualquiera sea el tipo de detector empleado, el mismo comenzará a proporcionar señal no en el instante preciso en que arriba el frente de onda, sino una fracción de período después. Esto introduce un error o **indeterminación**, que traducido en distancias es, justamente, una fracción de la

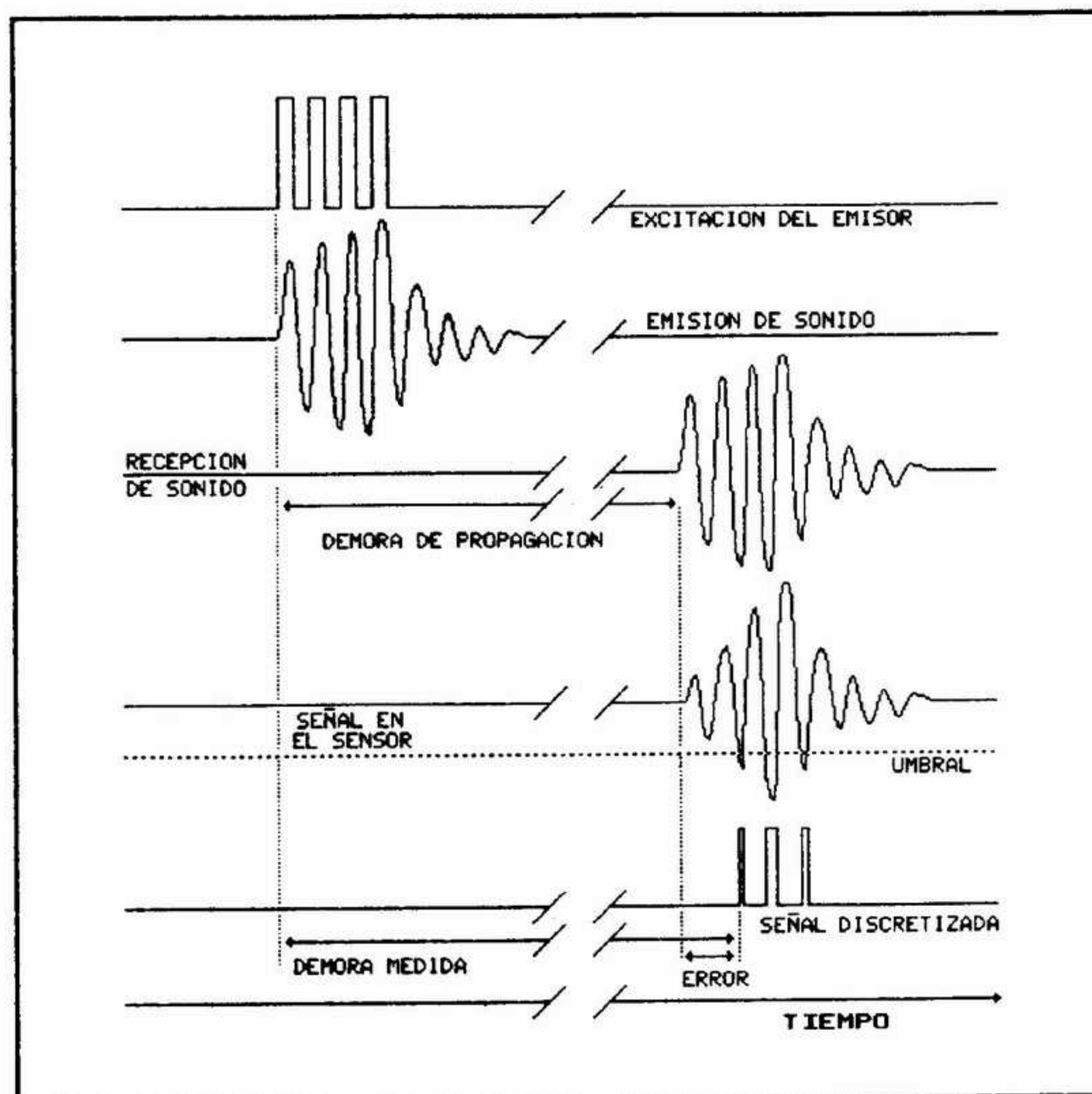


Fig. 2 - Formas de onda

longitud de onda λ . Si esta fracción fuese, por ejemplo, 0.2, a 1KHz la indeterminación resultaría de 6.6 cm.

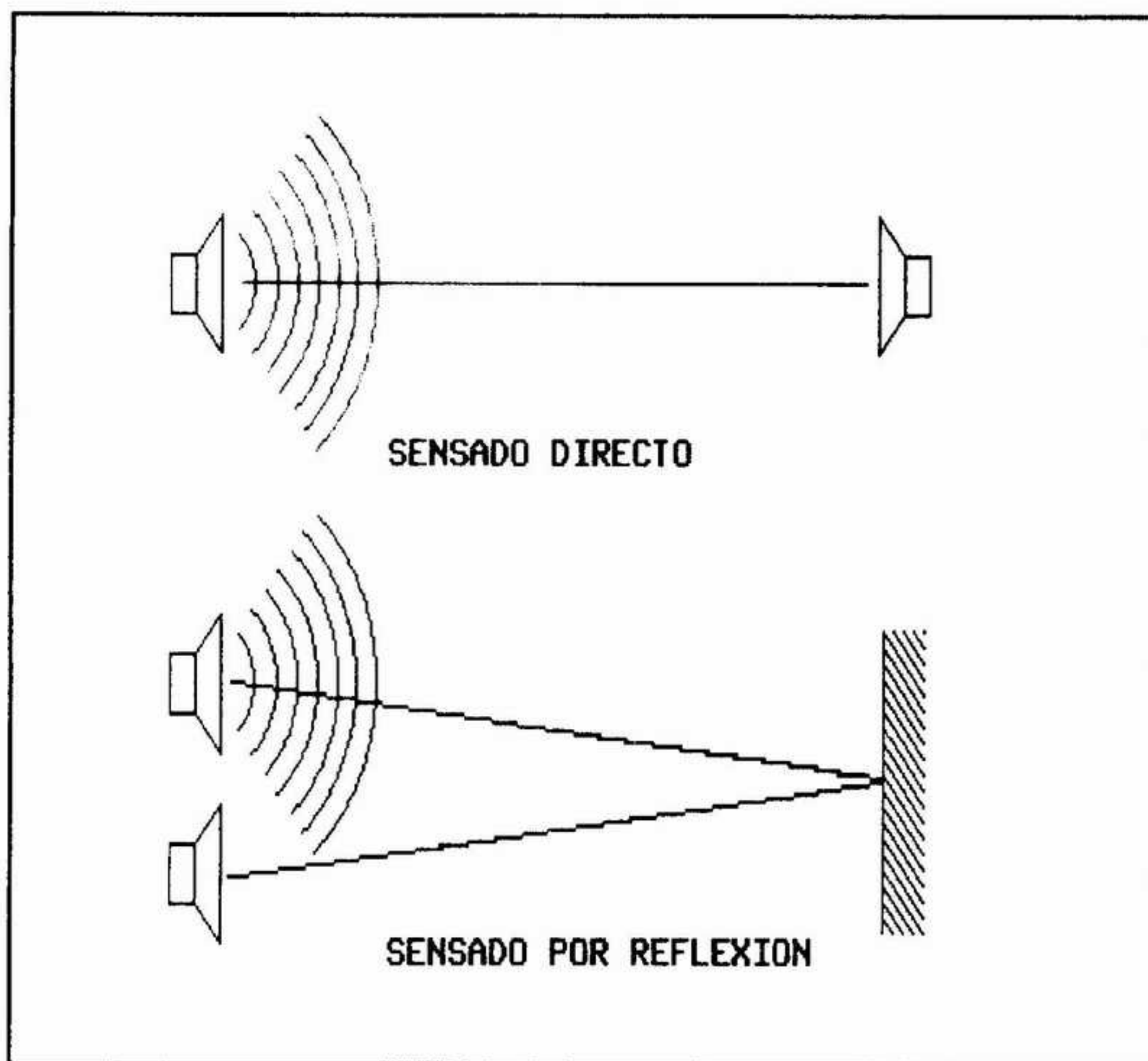


Fig. 3 - Sensado directo y por reflexión

Por todas estas razones es *menester emplear frecuencias tan elevadas como sea posible*, lo que normalmente conduce al uso de **ultrasonidos**: sonidos de frecuencia superior a la audible (más de 20 KHz).

Trasductores ultrasónicos

Para la generación y recepción de ultrasonido se emplean **trasductores** cerámicos, cuya respuesta a frecuencias es estrecha. Estos trasductores presentan un pico correspondiente a su frecuencia natural de resonancia, y normalmente se los emplea con esa misma frecuencia, que suele ser de 40 KHz.

Los trasductores se presentan como emisor y receptor separa-

dos, o bien como sensores duales, que pueden cumplir ambas funciones.

Una variante económica

A veces puede resultar difícil disponer de sensores ultrasónicos, y en todos los casos su costo puede ser restrictivo (sobre todo si lo que se desea es "jugar"). Una alternativa es emplear frecuencias menores, siempre del orden de varios KHz.

Para ello se pueden emplear trasductores cerámicos, de los comúnmente usados para buzzers, e incluso altavoces de pequeño diámetro.

En ambos casos se presenta una situación similar a la comentada anteriormente: la res-

puesta es particularmente acentuada para una frecuencia dada, que se deberá determinar experimentalmente, y luego utilizar.

Frecuencias características

Al contrario de lo que podría parecer a primera vista, la respuesta de un transductor (tanto en emisión como en sensado) *no es ni aproximadamente plana*. Se presentan picos de respuesta a determinadas frecuencia, que generalmente corresponden a la natural del transductor y sus múltiplos. Estas frecuencias se pueden determinar empíricamente, por prueba y error, o, más refinadamente, excitando el transductor y observando su respuesta con un osciloscopio. Un golpe, por ejemplo, causaría varios ciclos de oscilación a la frecuencia natural del transductor.

Emisión y sensado

Dos configuraciones típicas permiten medición de distancias. La primera resulta del empleo de un emisor y un receptor

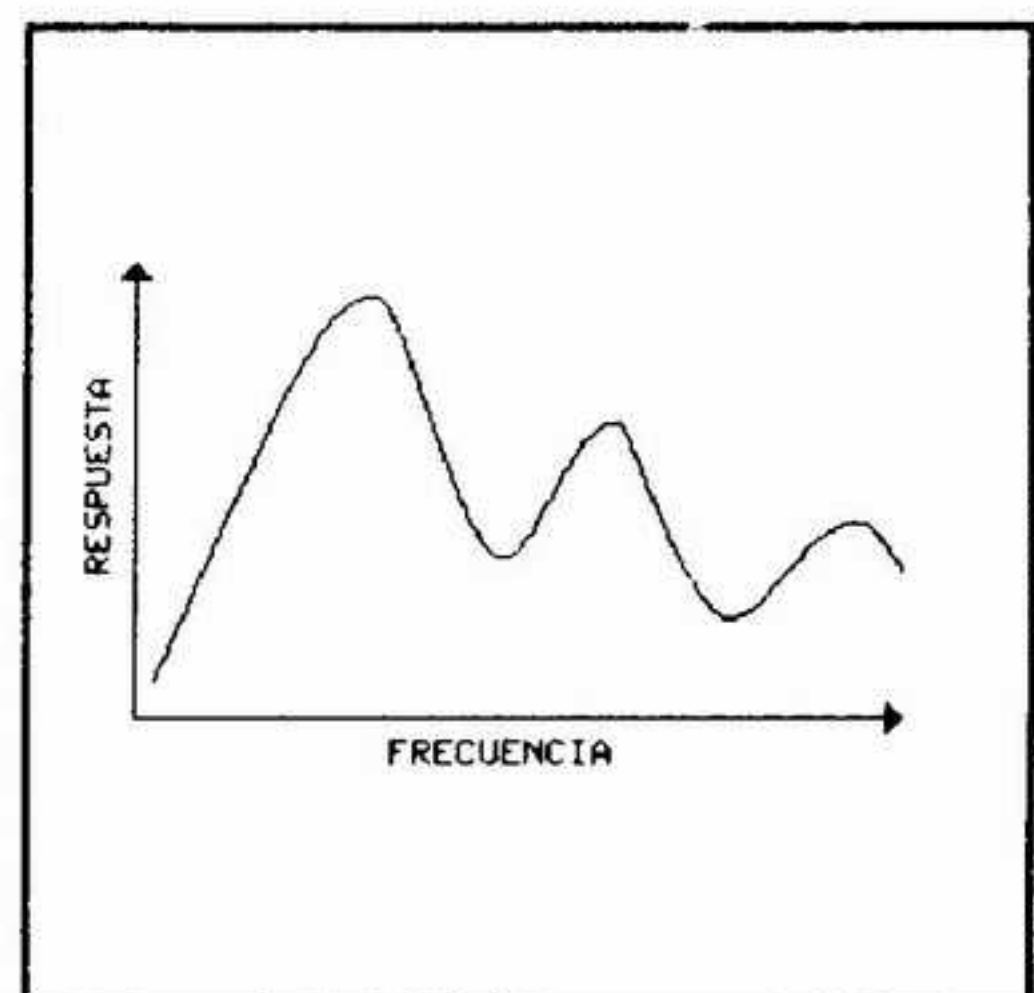


Fig. 4 - Las respuestas del sensor y del emisor presentan picos a diferentes frecuencias características

separados, que deben ser de características similares (para que coincidan sus frecuencias de resonancia y así exhiban una sensibilidad aceptable).

El emisor y el receptor pueden disponerse distanciados, y de esa forma el dispositivo medirá la distancia entre ambos, o bien juntos, sensando por reflexión.

La alternativa es emplear **un único transductor** como emisor y receptor, que entonces siempre operará por reflexión. Esta opción -no ilustrada en esta nota- es viable, de acuerdo con lo expuesto, *sólo si se opera a frecuencias en el rango de los ultrasonidos*.

Excitación

La excitación del emisor se consigue fácilmente desde una PC: basta inyectarle una onda cuadrada de la frecuencia deseada. La implementación práctica de esto se reduce a emplear un puerto de salida cualquiera y un transistor de uso general como excitador; el puerto se pulsa a la frecuencia predeterminada, fijada por un ciclo de demora.

Sensado

El sensado es un poco más complejo. Requiere, en primer término, amplificación de la señal, que es débil. Además, se deben suprimir las señales espurias (sonido ambiente y ruido eléctrico) que capta el sensor. Lo ideal aquí sería un filtro pasabanda centrado a la frecuencia de operación; en la práctica, la pobre respuesta del transduc-

```
defint a-z
do
    BitDato=1
    gosub Muestreo
    locate 1,1
    print Distancia!
loop

Muestreo:
    dim Muestra(10)
    Demora=300
    Factor!=0.3
    Port=&H378
    Prom0=0
    for Muestra0=1 to 7
        IMR=inp(&H21)
        out &H21,&Hff
        for i!=1 to 3000: next i!
        do
            Orig=inp(Port+1) and 8
            for i=1 to 3
                for j=1 to Demora:next j
                out port, BitDato
                for j=1 to Demora:next j
                out port, 0
            next i
            for i=1 to 1000
                if (inp(Port+1) and 8)<>Orig then exit for
            next i
            if i<=1000 then exit do
        loop
        Muestra(Muestra0)=i: Prom0=Prom0+i
    next Muestra0
    Prom0=Prom0/7
    out &H21,IMR
    Prom=0
    Muestras=0
    for Muestra0=1 to 7
        Relacion!=Muestra(Muestra0)/Prom0
        if Relacion!>.9 and Relacion!<.1 then
            Prom=Prom+Muestra(Muestra0)
            Muestras=Muestras+1
        end if
    next Muestra0
    Prom=Prom/Muestras
    Distancia!=Factor!*Prom
    return
```

Fig. 5 - Un programa para medición de distancias

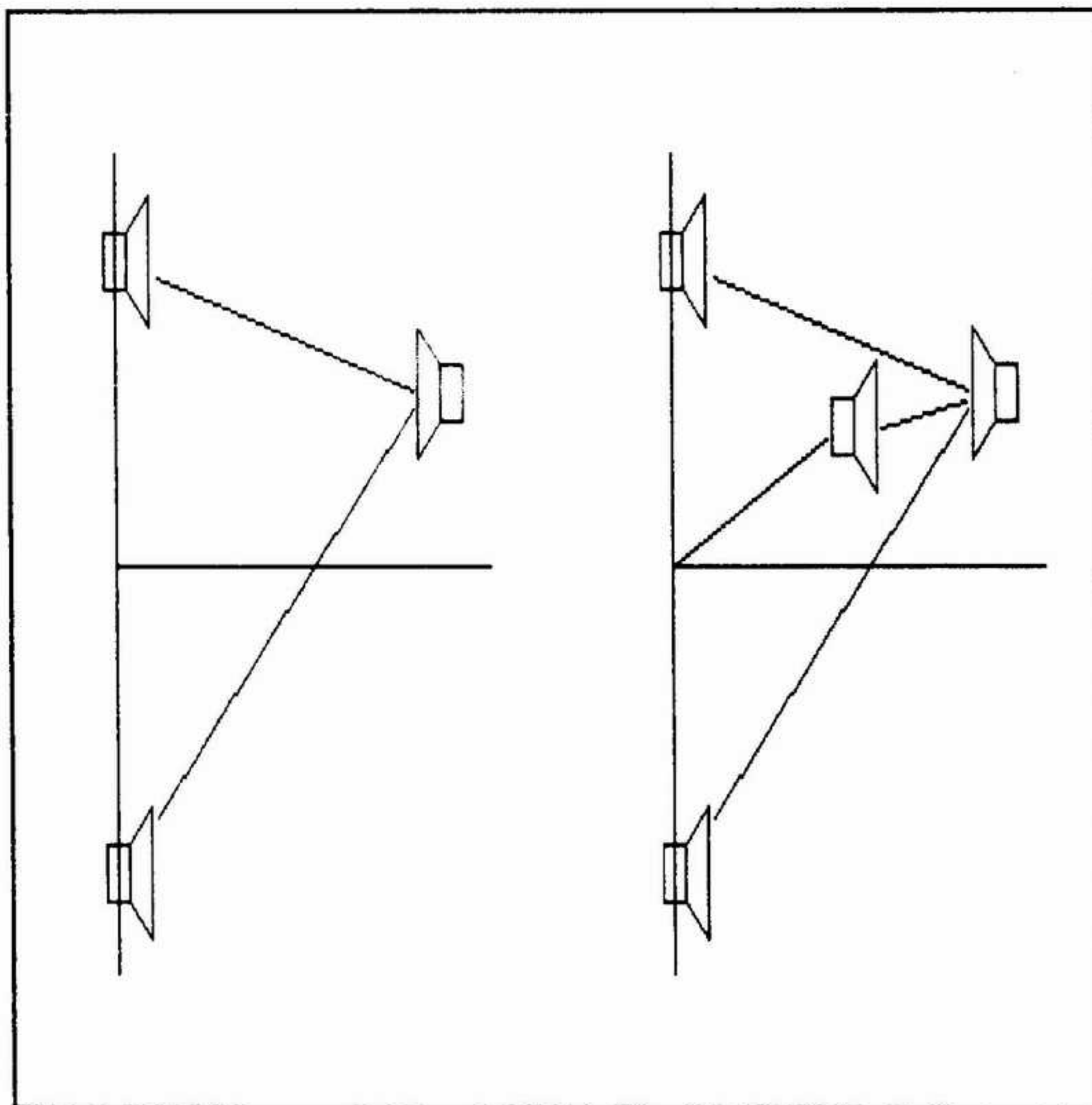


Fig. 6 - Disposición de dos o tres emisores para determinación de posición de otras tantas dimensiones

tor a bajas frecuencias puede ser suficiente. Con respecto al ruido, es imprescindible el empleo de cable blindado en la conexión del sensor.

Las componentes de alta frecuencia del ruido se filtran con un pasabajos; su frecuencia de corte, determinada por el capacitor de .1 μF debe adecuarse a la frecuencia de operación (en el esquema se supone una operación a 5 KHz).

El sensado culmina con la discriminación de nivel en la salida, realizada mediante una compuerta **Schmitt**. Una vez detectado un nivel sonoro suficiente, se activa la salida.

El ciclo completo

```

defint a-z
D!=30                                     ' Distancia entre emisores
do
    BitDato=1: gosub Muestreo: D1!=Distancia!
    BitDato=2: gosub Muestreo: D2!=Distancia!
    DC!=D!*D!: D1C!=D1!*D1!: D2C!=D2!*D2!
    Y!= (D2C!-D1C!)/2/D!
    X!= SQR((D2C!+D1C!)/2-Y!*Y!-DC!/4)
    locate 1,1
    print X!, Y!
loop
    
```

Fig. 7 : Con dos emisores actuados con DATA0 y DATA1 se pueden determinar las coordenadas del sensor en dos dimensiones

El ciclo de medición consta de tres fases. Durante la primera se excita el emisor con un breve **tren de pulsos** (normalmente bastan 3 a 5). Si se emplea un transductor único, a continuación se establece una pausa, durante la cual se suprime la excitación, y no se realiza sensado. Esta pausa garantiza que se estará permitiendo al sistema llegar a la condición de reposo. Mientras tanto, el tren emitido se propaga.

La tercera fase es el sensado, un ciclo de espera a que aparezca un nivel alto en el sensor. Obviamente, el tiempo transcurrido desde el instante de emisión hasta el de detección corresponde a la distancia recorrida, según:

$$D=T/V$$

donde V es la velocidad del sonido, 0.33 mm/ μs .

En disposiciones para sensado por reflexión, esta distancia co-

responde al trayecto completo de la onda sonora, que es el **doble** de la distancia entre el transductor y el elemento reflectante.

Como el tiempo de propagación determinado así está afectado por una indeterminación que puede ser de hasta algunos cm, y es posible que ocasionalmente se determinen -erróneamente- tiempos mucho más cortos debido a ruidos, es necesario **procesar varias muestras** para reducir las fluctuaciones en las distancias determinadas. El **factor de conversión** entre ciclos de espera y distancias depende de la velocidad del sonido, pero también de la velocidad de ejecución; es más simple determinarlo experimentalmente.

Como se vé, la implementación de este tipo de dispositivos es bastante sencilla. Vamos a considerar ahora algunas posibilidades interesantes de experimentación, empleando sensores múltiples.

Lo más práctico es, en este caso, usar un único detector y varios emisores, ya que la circuitería asociada a cada emisor es elementalmente sencilla.

Si en un plano se disponen dos emisores y un detector, y se determinan (separadamente) las distancias entre el último y cada uno de los emisores, es posible inferir la posición del detector en **dos dimensiones**. En realidad, subsiste una indeterminación: *de qué lado se encuentra el detector con respec-*

```
defint a-z
```

```
D!=30
```

```
' Distancia entre emisores
```

```
do
```

```
BitDato=1: gosub Muestreo: D1!=Distancia!
```

```
BitDato=2: gosub Muestreo: D2!=Distancia!
```

```
BitDato=4: gosub Muestreo: D3!=Distancia!
```

```
DC!=D1*D1: D1C!=D1!*D1!: D2C!=D2!*D2!: D3C!=D3!*D3!
```

```
Y!= (D2C!-D1C!)/2/D!
```

```
Z!= (2*D3C!-D1C!-D2C!-2*DC!)/4/D!
```

```
X!=SQR((D1C!+DC2!)/2-Z!*Z!-Y!*Y!-DC!/4)
```

```
locate 1,1
```

```
print X!, Y!
```

```
loop
```

Fig. 8 - Tres emisores, conectados a las líneas de datos 0, 1 y 2 permiten establecer la posición del sensor en TRES DIMENSIONES

to a la línea que une ambos emisores.

Esta indeterminación puede subsanarse empleando un tercer emisor, o bien restringiendo la zona de operación a un solo lado.

El programa de la figura 7 acciona alternativamente dos emisores separados por 30 cm, y determina las coordenadas del sensor. La posición del sensor se representa en pantalla; el dispositivo resulta, groseramente, un apuntador funcionalmente similar a un mouse.

Desde luego, la precisión no es la misma, pero este método tiene una ventaja respecto al mouse: no está confinado al plano.

3D

En efecto, podemos ampliar di-

mensionalmente este método de apuntado. Como se muestra en la figura 8, disponiendo tres (o cuatro) emisores, convertimos el apuntador en un dispositivo tridimensional.

No hay mucho software capaz de aceptar un apuntamiento de este estilo; pero es indudable que para algunas aplicaciones sería muy interesante.

Recordando al lector que la resolución de dispositivos de este tipo está ligada a la frecuencia con que se opere (y que, por lo tanto, a bajas frecuencias se tendrá una resolución muy pobre), lo invitamos a explorar este ámbito relativamente poco común, que puede sugerir otras aplicaciones de interés.



N o v e d a d e s e n s h a r e w a r e

Basics & Beyond

*por Pablo Díez **

**Se presenta aquí
una excelente guía
interactiva sobre
computación**

Si alguna vez quiso buscar información sobre temas técnicos de PC y nunca encontró la bibliografía adecuada, no busque más, el CD Rom Basics & Beyond resolverá todas sus dudas.

Con una interface gráfica que trabaja bajo sistemas Windows 3.1 o posteriores, se accede fácilmente a cualquiera de los temas contenidos, que están organizados en forma inteligente en tres niveles.

Debe elegir recorrer el programa entre los niveles básico, avanzado o experto, de manera tal que sacarán provecho desde usuarios novatos hasta los más expertos.

El menú principal es el punto de partida desde el cual se ingresa a una de las cuatro secciones principales del curso de aprendizaje. Se presentan los temas en una estructura arbolar, donde prácticamente se puede navegar en todas las direcciones

FICHA TECNICA:

TITULO: Basics & Beyond

FORMATO: CD Rom

ORIGEN: USA

REQUERIMIENTOS: 80386 SX o superior, 4 Mb de memoria RAM, monitor y placa gráfica Super VGA con una resolución de 640 x 480 x 256 colores, Windows 3.1 o superior y tarjeta de sonido.

PRECIO: \$ 45.- IVA incluido

DISTRIBUYE: Estudio Shareware

***(de Estudio Shareware)**

utilizando objetos "sensibles al mouse" (se iluminan o cambian de color cuando el apuntador se halla encima, o cambian de forma al presionar el botón), o bien usando los botones de control ubicados en la parte inferior de la pantalla.

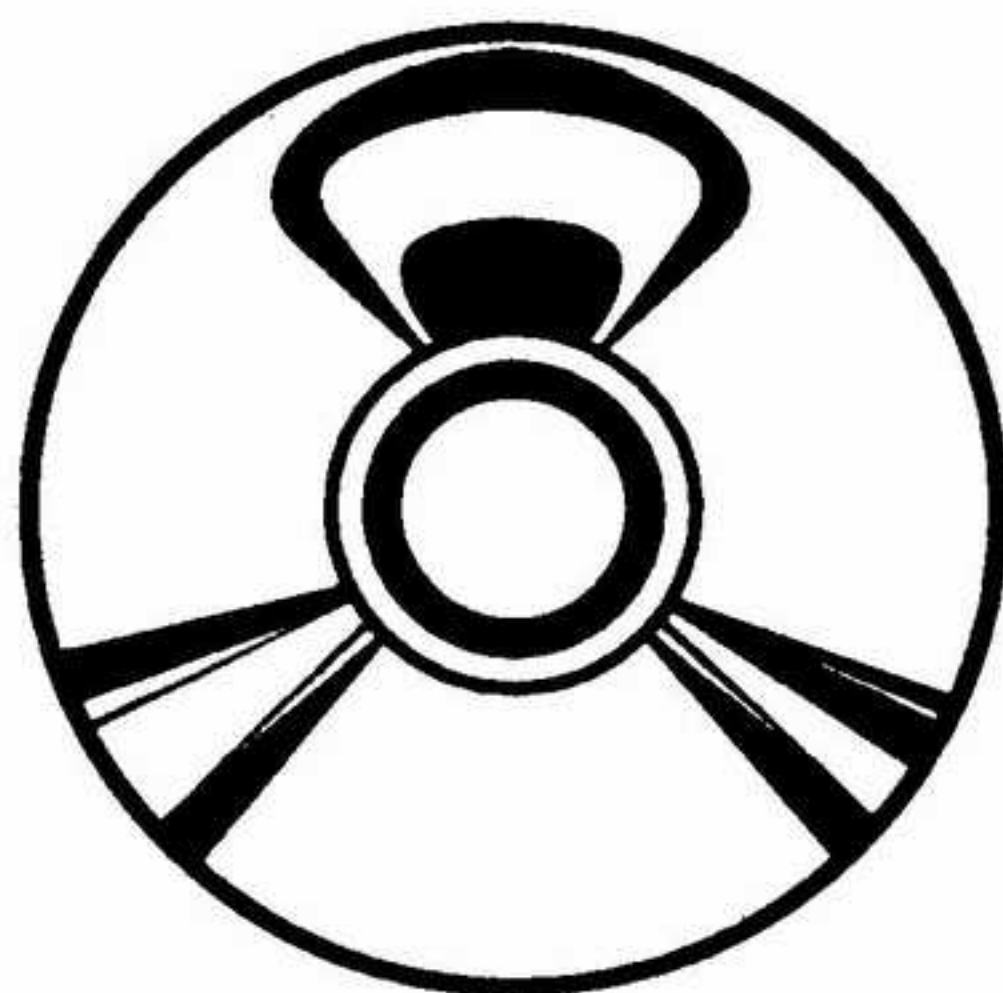
La información que se ofrece es del tipo "hipertexto", donde hay palabras o frases en letras rojas o negritas que se despliegan y derivan hacia otros temas relacionados con el actual, o aparece un pequeño cuadro explicando el significado de una palabra o frase.

Basics & Beyond explora más de 1000 temas técnicos, contiene más de 500 imágenes gráficas de alta resolución, con sonido, animación y video; incorporando un sistema de búsqueda por palabras clave, hipertexto, índice y ayuda sensible al contexto, que lo convierten en un manual de cabecera y referencia técnica único en su género.

El curso de aprendizaje está dividido en tres secciones principales a saber:

Básico: Se explican nociones introductorias desde la historia de la aparición de la PC, pasando por el "ABC" de la computación donde se explican términos y conceptos básicos que sirven para avanzar a niveles superiores, hasta configuración y mantenimiento de las máquinas.

Hardware: Contiene descripciones, definiciones y datos técnicos



nicos de los componentes, dispositivos y equipos que constituyen una PC. Puede introducirse en un componente, ya sea teclados, video, CPU, impresoras, periféricos o hardware para redes y profundizar en cada

uno de ellos desglosando los temas relacionados.

Software: Se divide en cuatro items donde se describen los sistemas operativos con sus características más importantes, los lenguajes de programación, los utilitarios y paquetes de aplicaciones comerciales, y finalmente los utilitarios de discos y antivirus.

Basics & Beyond también incluye un glosario especial, que contiene más de mil definiciones, sinónimos y temas relacionados.

GIGA SYSTEM BBS

20 Líneas rotativas - Modems de 14.400 bps

19 CDROMS en línea

Protocolo gráfico - Teleconferencia

400 MB SHARE nuevos todos los meses

FIDONET - INTERNET

140 Foros de alcance Nacional e Internacional

Red Educativa - Areas Empresariales

COMUNICACION AL INSTANTE

Setee su Modem en 8N1 y disque 702.1072 - 703.1073

Consultas y soporte técnico al 703.2289



Medios tonos

Fundamentos y métodos para la producción de medios tonos

La representación de imágenes a través de diferentes medios requiere -según el grado de realismo y las características cromáticas del motivo- una gama más o menos extensa de colores.

Sin embargo, lo común es que la paleta de colores puros disponibles sea siempre mucho más limitada que los requerimientos, por modestos que ellos sean.

Así, disponiendo de una impresora en blanco y negro, llegará el momento en que se necesite imprimir imágenes con una gama de grises de latitud importante, o que se precise una paleta de 16 colores cuando sólo cuatro tonos primarios sean los disponibles.

Yendo todavía más lejos, es usual manejar paletas de 256 colores basadas en 16 tonos disponibles, y aún esa cantidad parece insuficiente.

Los medios digitales presentan

limitaciones inherentes en cuanto a la cantidad de colores primarios; esas limitaciones se encuentran también en otros ámbitos (por ejemplo, la gráfica: la cantidad de tintas es también limitada, y cada una implica un paso adicional de impresión).

Naturalmente, estos problemas son salvados acudiendo a combinaciones aditivas o sustractivas de colores, dispuestos en pequeños puntos de los que se espera que resulten individualmente imperceptibles.

Medios tonos

El caso extremo de esta restricción se encuentra en los dispositivos que sólo pueden presentar **dos estados** (dos colores): monitores (o modos de video) monocromos, la gran parte de las impresoras, y la mayoría del equipo de digitización. Desde luego, se tratará siempre de imágenes monocromáticas, pero de alguna forma se debe poder manejar una escala de to-



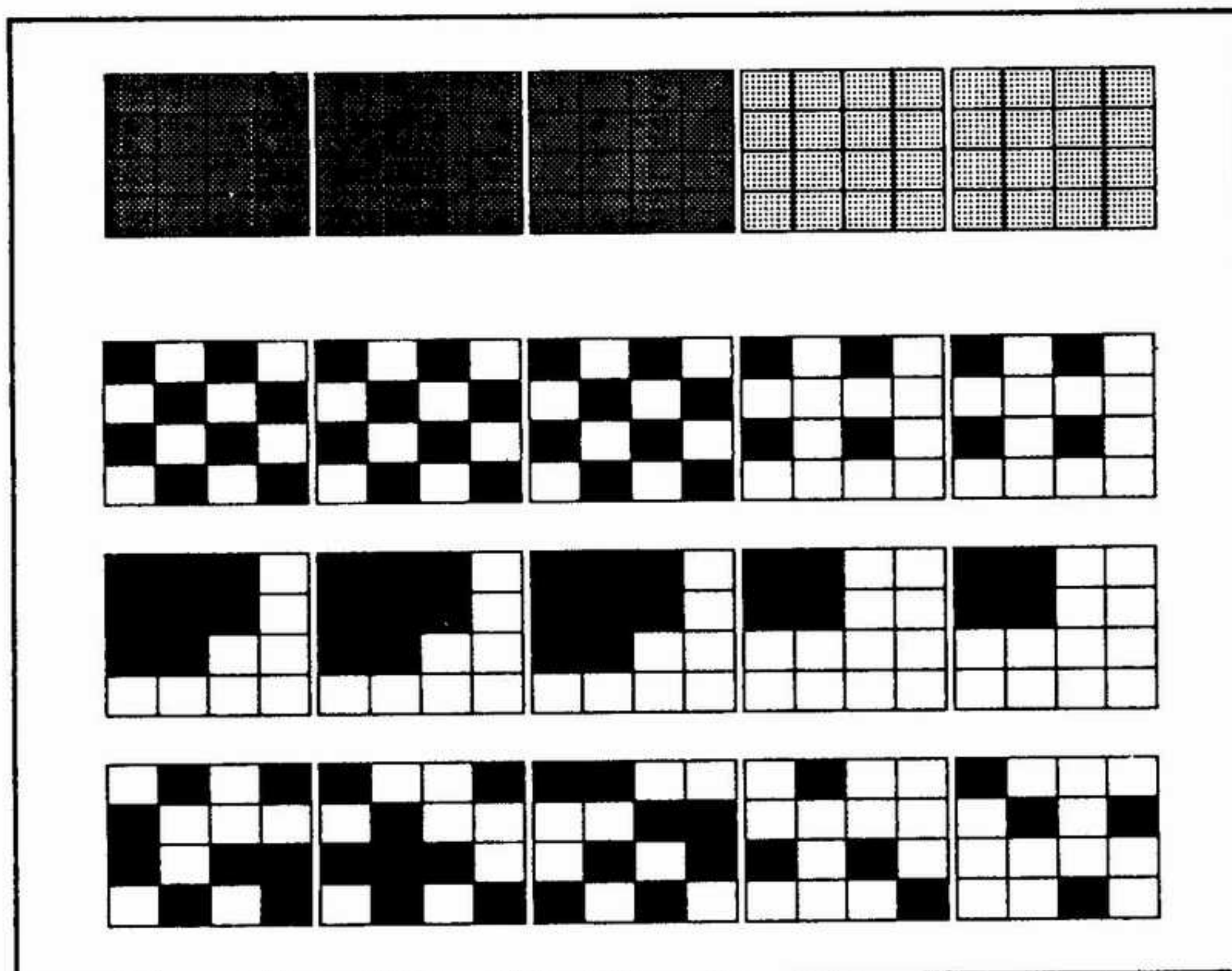


Fig. 1 - Medios tonos en celdas de 4x4, correspondientes a los niveles de gris de la porción superior. Se ilustran los métodos de Bayer, engrosamiento de punto y difusión

nos, aunque sólo se trate de una apariencia subjetiva.

Como en todo lo que en esta nota se discute, el limitante final es casi siempre la *impresión subjetiva*. Vimos que la percep-

ción visual posibilita "falsificar" grises mediante pequeños puntos negros sobre fondo claro, o al revés. No sólo se aprovecha este hecho: se lo explota hasta sus límites más lejanos.

Las técnicas de medio tono buscan obtener una impresión subjetiva de gris empleando puntos negros o blancos de diferente tamaño o densidad, según lo permita el dispositivo. Monitores, scanners e impresoras de matriz sólo permiten un tamaño de punto (u, obviamente, tamaños múltiplo). Algunas impresoras láser permiten varios tamaños de punto o una modulación cuasi continua.

Si el medio tono se basara en el uso de puntos *por debajo* del poder de resolución de la vista (como algunos creen), la elección de una técnica descansaría sólo en performance y no en resultados. En muchos casos, los puntos *sí son discernibles*, pero se los dispone de manera que no atraigan la atención.

Algorítmicamente, las técnicas no son complejas (por lo menos si no se exige una alta performance). En todos los casos se

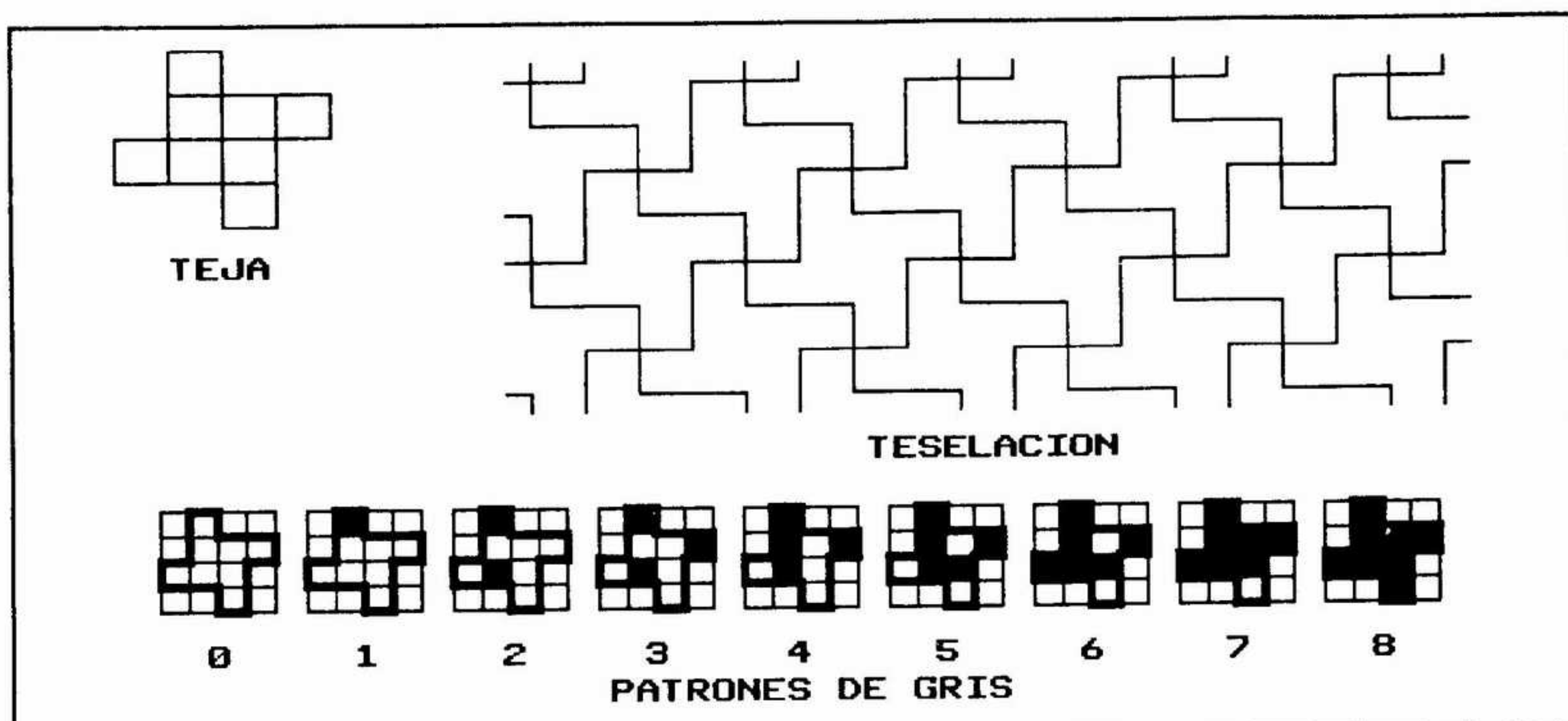


Fig. 2 - Teselación de una celda de 8 pixels y codificación de los nueve niveles de gris resultantes

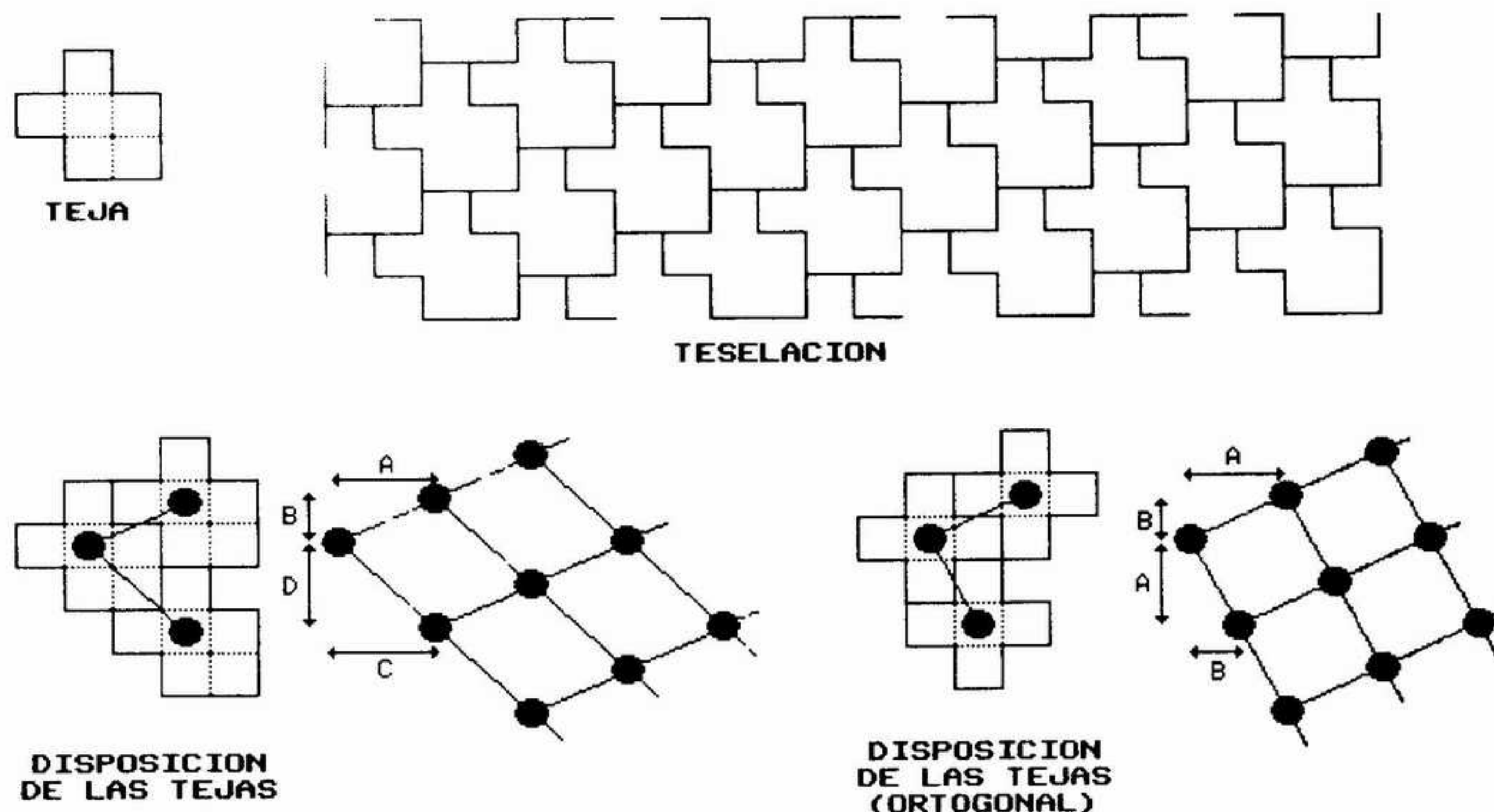


Fig. 3 - Disposición de las celdas en una teselación, mostrando el ángulo de la trama. A la derecha, una trama ortogonal.

define un algoritmo capaz de convertir un tono de gris (de una gama de N tonos posibles) a una trama punteada. Veamos los algoritmos, pero desde el punto de vista de sus fundamentos.

B/N y color

Las técnicas para generar medios tonos se basan en que el ojo promediará el valor cromático de puntos muy próximos, cuando el tamaño del conjunto está cercano al que permite discernir la agudeza visual. Esto es igualmente válido cuando se

trata de grises y cuando se maneja color.

En cualquier caso, el valor cromático aparente resulta de una ley de combinación, por adición o sustracción: un promedio ponderado de los diferentes valores cromáticos, con pesos proporcionales a la frecuencia o proporción de puntos de cada color.

En el caso de los grises, el valor cromático es único, y bastan dos tonos de gris diferentes para obtener por combinación los intermedios.

Para las imágenes cromáticas, el problema es esencialmente el mismo, excepto que el valor de color viene dado por una **terna** de cantidades, y en el caso más general se exigirá un combinación de cuatro tonos.

Es así que la mayor complejidad de este último caso reside exclusivamente en una mayor dificultad para elegir los componentes primarios y determinar su combinación, aspecto que trataremos en la próxima nota al respecto. Por lo demás, todo lo expuesto en la presente, re-

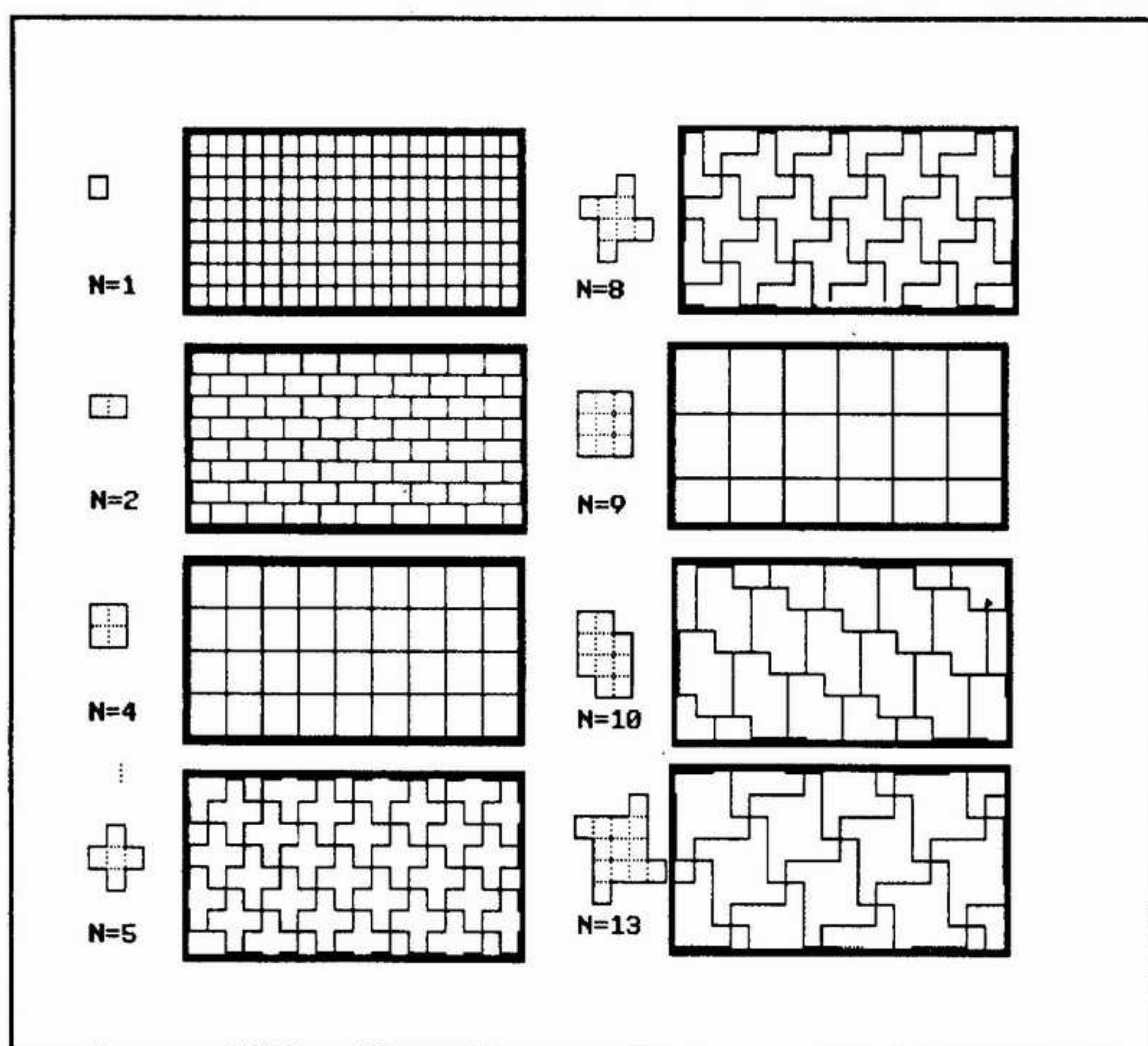


Fig. 4 - Teselaciones ortogonales. Para celdas de 13 o menos píxels, las presentadas son las únicas posibles

ferido a grises, es aplicable a medios tonos en color.

En lo sucesivo, vamos a suponer que se dispone de una imagen codificada píxel a píxel, con una resolución **igual** a la del dispositivo de presentación (por ejemplo, 300 dpi para una impresora láser). Como ejemplo, supondremos uniformemente que se trata de una imagen destinada a ser presentada en un monitor, con lo que cualquier ensayo se podrá llevar a cabo con facilidad.

En cuanto al valor de cada píxel, asumimos que se trata de un entero comprendido entre cero y un máximo, determinando una gama de N grises.

Antes de considerar algunos fundamentos básicos, veamos las dos técnicas más sencillas para representar esta gama.

Umbral

Citamos esta técnica aquí por tratarse del método más elemental de convertir una imagen multitonal a una de sólo dos tonos (o, en general, de N tonos). Consiste, simplemente, en **discretizar** el color en blanco o negro, y sólo tiene sentido si se pretende obtener siluetas, efectuar pruebas o pasar a un **line art**. Simplemente se define un valor **umbral** del gris; por debajo de él, se codifica negro, y por encima, blanco. En el caso de imágenes cromáticas, cada tono posible se sustituye con el color puro más parecido.

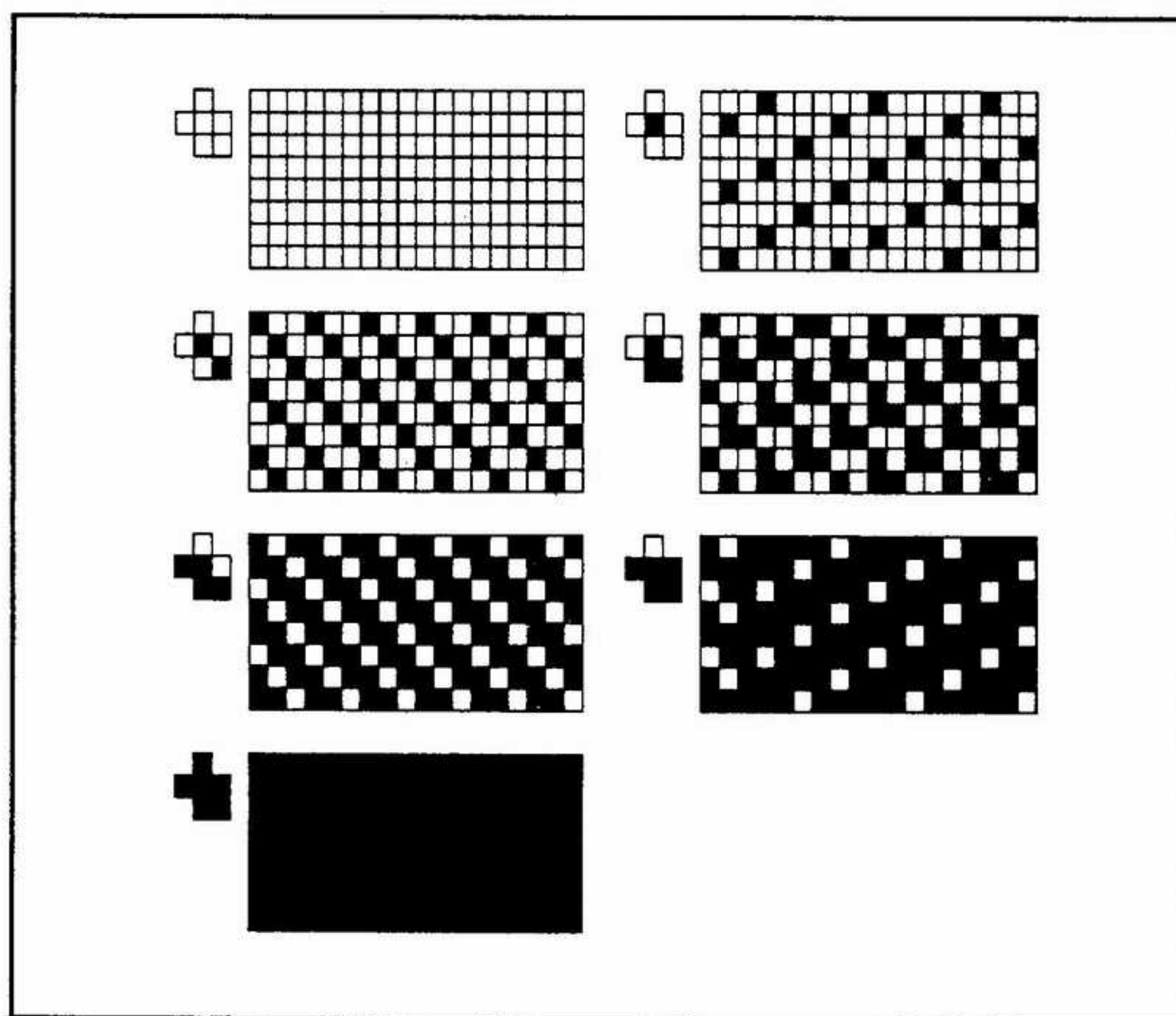


Fig. 5 - Siete tonos de gris para una celda de 6 píxels

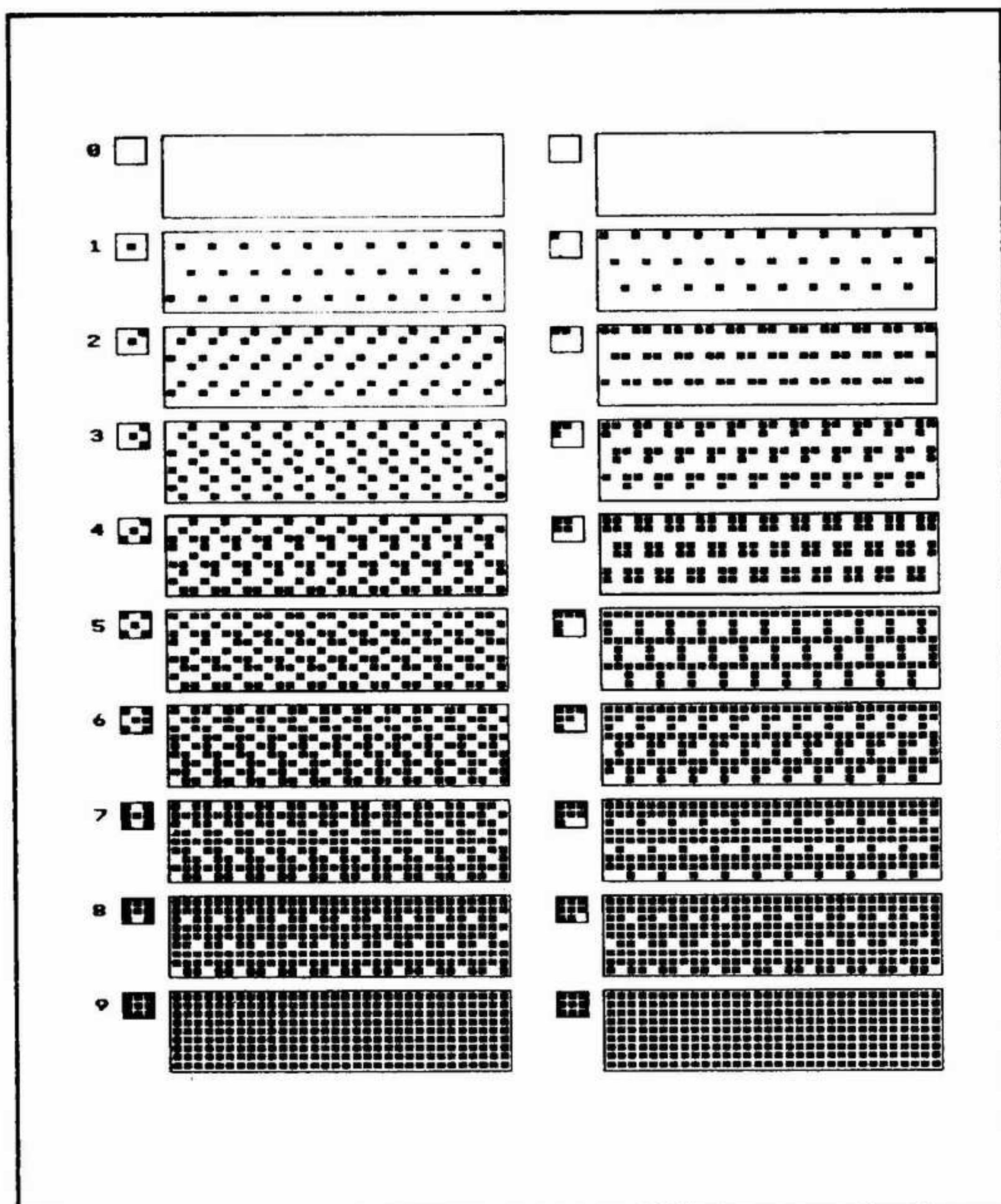


Fig. 6 - A la izquierda, la dispersión de los puntos se maximiza (Bayer) A la derecha, los puntos se agrupan de manera de obtener spots de diferente tamaño

Difusión

Esta se trata de una aproximación estadística: cada píxel se transforma a un punto cuya **probabilidad** de ser blanco es proporcional al valor del tono.

En la práctica, se genera un **valor de prueba** al azar, y si el valor del tono es superior se deja el punto en blanco; negro en caso contrario.

El método de difusión es fácil de implementar, y da buenos resultados cuando la imagen pre-

senta mayoritariamente áreas de tonos continuamente variable, más que zonas de color más o menos plano, porque no se advierten discontinuidades en la coloración aparente.

En las áreas planas, particularmente las casi negras o casi blancas, resulta muy evidente el punteado.

La extensión de la gama de grises obtenida depende de la resolución de la salida y de las condiciones de observación, lo

que se apreciará mejor luego de considerar las técnicas restantes.

Celdas

La otra aproximación al problema se basa en considerar la relación entre la resolución del dispositivo, y la resolución visual del observador en determinadas condiciones. Primero vamos a suponer que esta última es fija, y que el observador no puede distinguir puntos separados en menos de N píxels. Bastaría entonces dividir la imagen en celdas de $N \times N$ píxels y colorear en cada celda un número de puntos **proporcional** al valor cromático promedio de los píxels involucrados. Si $A = N \times N$ es la cantidad de píxels de la celda, el número de colores disponibles será de $A+1$.

La realidad es que la resolución visual del observador depende de muchos factores, entre ellos, *algunos de tipo subjetivo*. Se establece así un compromiso entre **cantidad de tonos de gris** (lineal con el tamaño o área de la celda) y la **resolución** de la imagen medida ya no en puntos sino en **celdas** por unidad de longitud.

Esta medida (expresada usualmente como **spi -spots per inch-**) es groseramente igual a la resolución dividida por A .

Como para un algoritmo dado (y para un determinado tono) la posición de los puntos coloreados en la celda es siempre la misma, en un área lisa aparecerá un **patrón** periódico de pun-

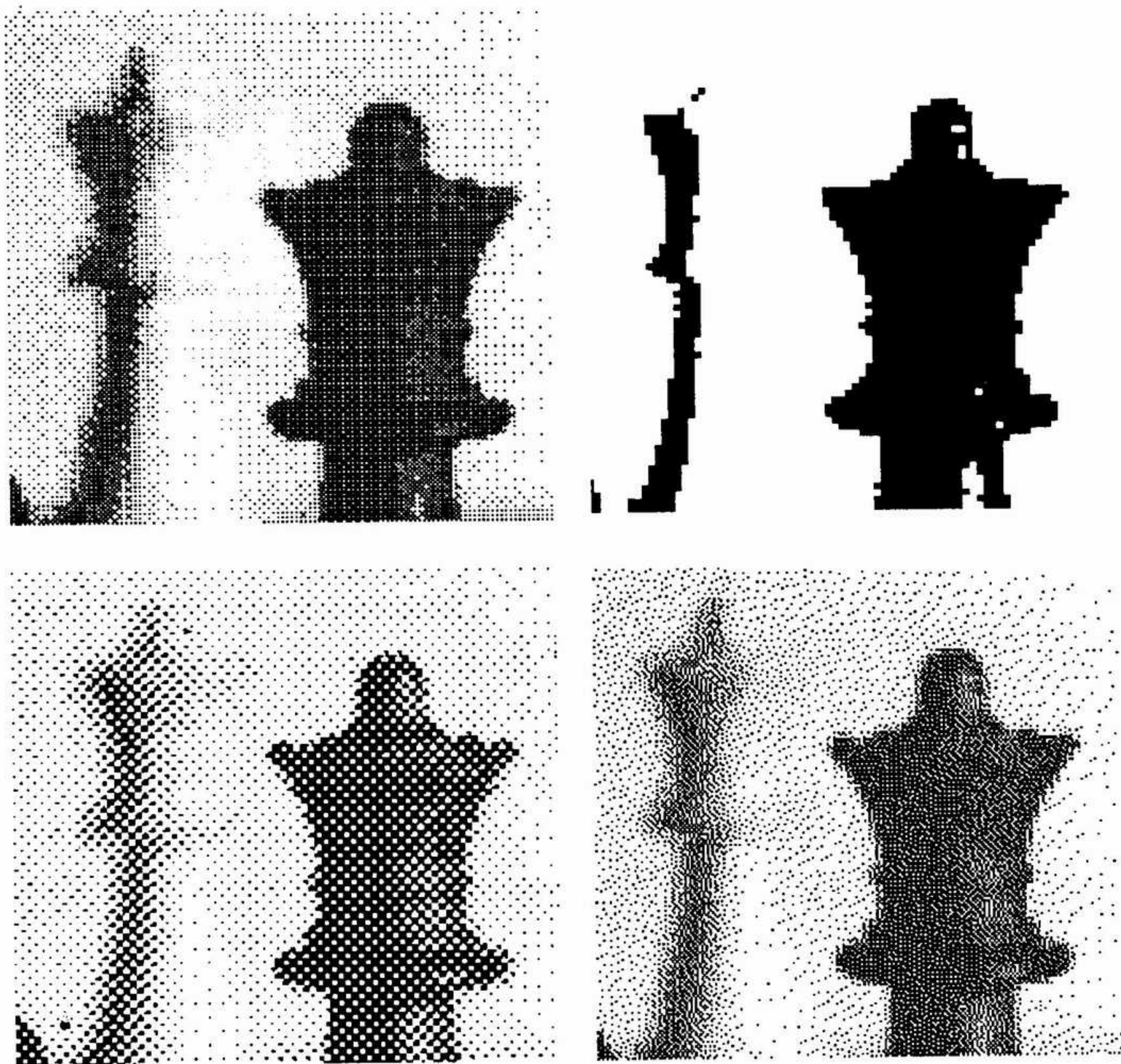


Fig. 7 - El resultado de la aplicación de los diferentes métodos a una misma imagen. Arriba, a la izquierda, puntos de tamaño variable; a derecha, discriminación por umbral. Abajo: Bayer (izquierda) y difusión (derecha). El tamaño de celda ha sido exagerado.

tos. La percepción de patrones tales depende fuertemente de la dirección de su **orientación**, siendo mucho más sensible a alineaciones próximas a la horizontal o a la vertical.

Además, la mayor parte de las imágenes suele presentar elementos casi verticales o casi horizontales, cuya presentación en medios tonos con cel-

das cuadradas tiende a producir efectos indeseables (*jagging*, o aserrado de las líneas y bordes).

La consecuencia es que en la mayoría de los casos se busca evitar la ocurrencia de patrones verticales u horizontales, acudiendo a otros tipos de celdas.

Teselación (Tiling)

La primera condición es que, obviamente, un conjunto de celdas iguales debe poder llenar el plano, sin vacíos ni superposiciones. Una teselación (tiling) o mosaico es una yuxtaposición de figuras congruentes que coinciden borde a borde llenando el plano (el lector puede pensar en cualquier tipo de mosai-


```

defint a-z
screen 9
x$=input$(1)
dim Orden1(3,3), Orden2(3,3)

DATA 1, 9, 2, 11
DATA 14, 5, 15, 7
DATA 4, 12, 3, 10
DATA 16, 8, 13, 6

for i=0 to 3
  for j=0 to 3
    read Orden1(i, j)
  next j
next i

DATA 1, 2, 5, 7
DATA 3, 4, 8, 11
DATA 6, 9, 13, 14
DATA 10, 12, 15, 16

for i=0 to 3
  for j=0 to 3
    read Orden2(i, j)
  next j
next i

for R0=0 to 349 step 4
  for C0=0 to 639 step 4
    ClrProm!=0
    for R=0 to 3
      for C=0 to 3
        ClrProm!=ClrProm!+point(C0+C,R0+R)
      next C
    next R
    ClrProm!=ClrProm!/4/4
    for R=0 to 3
      for C=0 to 3
        gosub HalfTone1
        pset(C0+C,R0+R), Clr
      next C
    next R
  next C0
next R0
end
HalfTone1:
  if Orden1(R,C)<=ClrProm! then Clr=15 else Clr=0
  return
HalfTone2:
  if Orden2(R,C)<=ClrProm! then Clr=15 else Clr=0
  return
HalfTone3:
  if ClrProm!>=8 then Clr=15 else Clr=0
  return
HalfTone4: if RND*15!<=ClrProm! then Clr=15 else Clr=0
  return

```

Fig. 8 - Conversión a medios tonos

co, mayólica o parqué, aún de forma irregular, pero le recomiendo que admire los hermosos -y numerosos- grabados del holandés **Escher**, basados en teselaciones con figuras complicadas).

Propuesto un número A de píxels por celda, **sólo algunas configuraciones admiten la teselación**. Pero aún hay otras restricciones.

La primera, es que la celda debe ser poseer una relación de alto a ancho más o menos proporcionada; la segunda, que la trama de celdas debe guardar distancias similares entre celdas vecinas (porque de otro modo, la resolución en un sentido será muy diferente a la medida en el otro sentido de la trama).

En la figura 2 se muestra una teselación obtenida a partir de una celda de cinco píxels; la siguiente figura muestra cómo se disponen dos tipos de celdas. Se observa que, si se consideran celdas vecinas, estas se alinean formando un determinado **ángulo** con la horizontal (que eventualmente puede coincidir con ella, o serle perpendicular).

En un sentido, cada celda se encuentra desplazada en **a** unidades horizontales y **b** verticales respecto de la precedente; en el restante, el desplazamiento es de **c** y **d** unidades respectivamente.

Se puede deducir que el área de los paralelogramos trazados

en la figura 3 *debe ser necesariamente igual a A*, el área de la celda, y que por lo tanto debe ser:

$$a \cdot d + b \cdot c = A$$

Como **a**, **b**, **c**, **d** son enteros pequeños, esto limita las posibilidades de teselación.

Una limitante adicional se encuentra si se pretende una teselación "densa", con un **ángulo recto** entre ambos sentidos de la trama (ortogonal).

En este caso resulta:

$$a = d$$

$$b = c$$

y por lo tanto debe ser:

$$a^2 + b^2 = A$$

Esta nueva restricción nos indica que **A no puede tomar cualquier valor**: debe poder descomponerse en dos cuadrados (uno de los cuales puede ser 0², en cuyo caso se tiene una trama recta). Así, se pueden obtener teselaciones densas con celdas de 1, 2, 4, o 5 píxels, pero no para 3, 6 o 7.

El ángulo α de la trama viene dado por:

$$\text{tg}(\alpha) = b/a$$

Una vez escogida una geometría de teja de A píxels, cada una de ellas admite A+1 valores aparentes de gris, según la cantidad de píxels de la teja que se encuentren en blanco o en negro.

Para ejemplificar el método recurrimos a la figura 2, que ilustra una teja, la forma en que

puede llenar el plano, y una posible gama de colores (grises) aparentes.

Cualquiera que sea la forma de la teja, es obvio que cuantos más tonos admita, menos tejas por unidad de superficie habrá. La cantidad de tejas por unidad de superficie será igual a la resolución (medida en píxels) dividida por la cantidad de píxels por teja.

Por ejemplo, para una resolución final de 300 dpi en ambos sentidos, horizontal y vertical, si se proponen 10 tonos de gris se requerirán tejas de 9 píxels, y una pulgada cuadrada contendrá 10000. Linealmente, esto corresponde a una resolución de 100 tejas por pulgada, o 100 SPI (spots per inch).

Para 37 colores, cada teja contendrá 36 píxels y la resolución descenderá a 50 spi.

Como se ve, existe un compromiso entre la amplitud de la gama de grises y la resolución: con altas resoluciones, la imagen se solariza, mostrando evidentes saltos de tonalidad; para gamas de numerosos tonos, el aspecto de mosaico se hace aparente.

Tejas de 8 por 8

En el caso de video, debido a la penalidad que pesa sobre cada acceso, es frecuente usar tejas de 8x8, o fracciones de ellas.

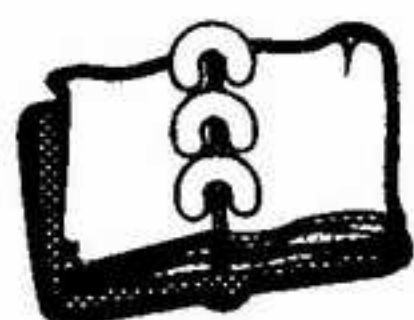
La razón estriba en que mientras que una imagen se compone de píxels, la memoria de video se maneja en bytes de 2, 4

u 8 píxels. El acceso a píxels individuales es muy lento en comparación con el acceso a bytes completos, y además la VGA permite el uso de una máscara muy apropiada para escribir patrones de un byte de ancho.

Sin embargo, en este artículo, los algoritmos propuestos operan píxel a píxel, en aras de la interpretabilidad. Desde luego, esto no es lo óptimo en lo que se refiere a velocidad. Dejamos al lector la tarea (que será sin duda instructiva e interesante, en el caso de buscar performance) de readaptarlos para manejo simultáneo de 8 píxels.

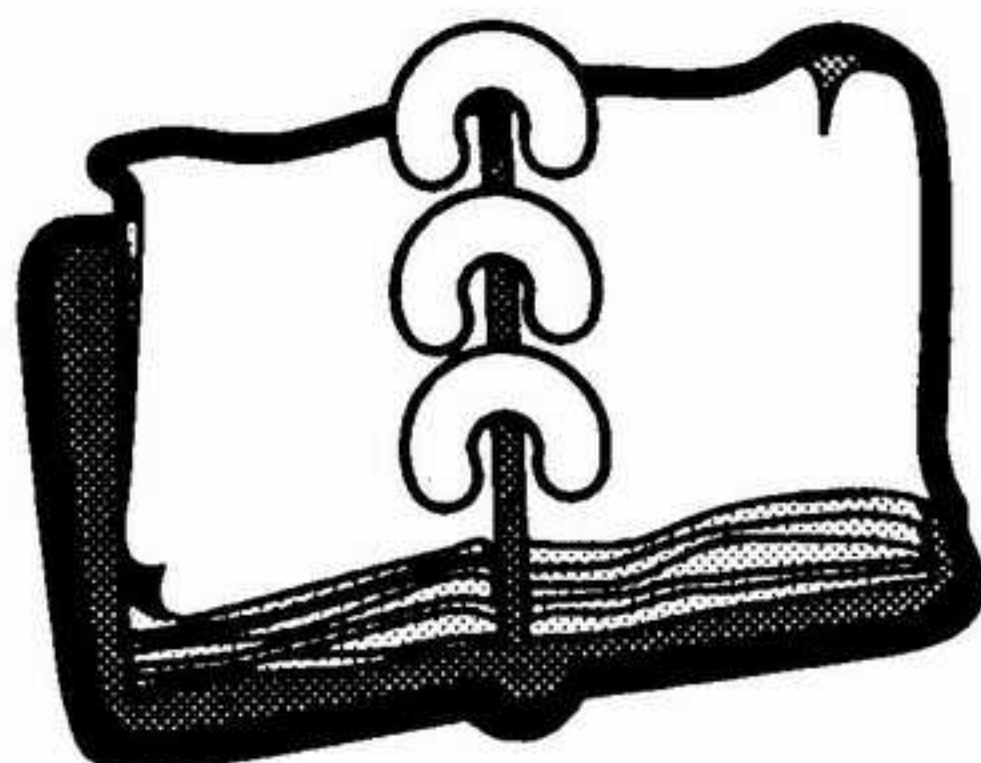
El programa de la figura 8 ejemplifica lo expuesto, procesando el contenido presentado por la pantalla al comenzar (supuestamente en 16 tonos de gris) y convirtiéndolo a medios tonos construidos sólo con blanco y negro. Al igual que en la nota sobre reconocimiento de patrones presentada en este mismo número, se sugiere emplear un capturador de pantalla como Frieze para cargar la imagen a procesar en el momento oportuno.

Las diferentes técnicas se implementan mediante otras tantas subrutinas, sobre una celda cuadrada de 4x4 (por motivos de simplicidad). Cuando se trata de patrones repetitivos, el orden de llenado de los puntos se definen inicialmente en las matrices **Orden1** y **Orden2**, cuyos ejemplos corresponden a las dos técnicas consideradas.



Acceso a disco desde TSRs

El acceso
asincrónico a
disco desde
residentes se
topa con un
problema
sistemático:
DOS no es
reentrante



Un problema típico se presenta en TSRs que deben acceder a archivos en disco: como el requerimiento de acceso puede -en principio- producirse *en cualquier momento*, inevitablemente terminará por ocurrir en un instante en que DOS está procesando otro acceso (o alguna otra función interna). Como DOS **no es reentrante**, esta situación derivará finalmente en un **crash**.

Tan usual es este problema, que su sola mención en nuestro **Correo de Lectores** desató un aluvión de consultas de parte de interesados en resolverlo con distintas finalidades; como de costumbre, intentamos que **PC Práctica** responda a esas inquietudes, y en consecuencia, aquí va esta nota.

Reentrada

En un diagrama de flujo clásico, el seguimiento de la lógica es

bastante simple: basta analizar cada condición de bifurcación y seguir la senda determinada. En algunos casos puede ocurrir que un procedimiento -directa o indirectamente- termine por invocarse a sí mismo, y quizás con parámetros que difieran de los de la invocación original. Esta reentrada al procedimiento puede tener dos consecuencias.

Si cada vez que se ingresa al procedimiento se emplea un nuevo juego de variables, todo funcionará normalmente. Esto ocurre, en general, para procedimientos cuyas variables son **dinámicas** (es decir, se alojan en el stack): cada invocación crea un nuevo set de ellas.

En el caso contrario (procedimientos donde alguna o todas las variables son estáticas), el valor de esas variables almacenado durante la primera invoca-


```

TRUE     EQU 0FFh
FALSE    EQU 000h

CODE     SEGMENT
ASSUME CS: CODE, DS: NOTHING
ORG      100h
JMP      INSTALAR

INCLUDE HOOKDOS.ASM
INCLUDE FILEOPS.ASM
INCLUDE BUFFER.ASM

ASSUME CS: CODE, DS: NOTHING
VectorPrt label dword
VectorPrtL label word
        dw 0
VectorPrtH label word
        dw 0
ServicioPrt proc far
        cmp ah, 0
        jnz PrtStat
        cmp al, 1Ah
        jnz PrtBuffer
        call FlushBuffer
        jmp PrtStat
PrtBuffer:
        call WriteBuffer
PrtStat:
        mov ah, 90h
        iret
ServicioPrt endp

Instalar:
        call HookDOS
        mov cs:Request, FALSE
        mov cs:BufferPtr, offset Buffer1

        xor ax, ax
        mov ds, ax
        mov bx, 17h*4
        mov ax, [bx]
        mov VectorPrtL, ax
        mov [bx], offset ServicioPrt
        mov ax, [bx+2]
        mov VectorPrtH, ax
        mov [bx+2], cs
        push cs
        pop ds
        mov dx, offset Instalar
        shr dx, 1
        shr dx, 1
        shr dx, 1
        shr dx, 1
        inc dx
        mov ax, 3100h
        call Int21

CODE ENDS
END

```

Fig. 1 - Cuerpo principal del programa y procedimiento de instalación

ción es corrompido por la operación en la segunda entrada, y el resultado es un *crash*. Los procedimientos que admiten la reentrada se denominan **reen- trantes**.

Interrupciones

Normalmente basta analizar el diagrama de flujo para asegurarse de que ningún procedimiento no reentrante se esté *autoinvocando*. Este simple análisis se complica cuando entran en juego las **interrupciones**: entonces el flujo deja de ser el visible en el papel, porque eventualmente una interrupción puede disparar procesos en cualquier momento.

DOS no es reentrante

A pesar de que el universo PC está constelado de residentes, DOS siempre fue un operativo **no reentrante**. La colisión a la que referimos al iniciar esta nota se produce, típicamente, cuando se está ejecutando una función de DOS, sobreviene una interrupción, y el programa que toma control a causa de la interrupción intenta emplear esa -u otra- función de DOS.

Casos

La necesidad de acceder asincrónicamente a archivos aparece en numerosos casos: además de residentes elaborados (cuyo ejemplo clásico y de "la primera hora" es **SideKick**), spoolers, capturadores de pan-

Vector21	label	dword	; Vector original de int 21h
Vector21L	label	word	
dw	0		
Vector21H	label	word	
dw	0		
Flag21	label	byte	; Flag de int 21h en curso
db	0		
db	0		
Vector25	label	dword	; Vector original de int 25h
Vector25L	label	word	
dw	0		
Vector25H	label	word	
dw	0		
Flag25	label	byte	; Flag de int 25h en curso
db	0		
db	0		
Vector26	label	dword	; Vector original de int 26h
Vector26L	label	word	
dw	0		
Vector26H	label	word	
dw	0		
Flag26	label	byte	; Flag de int 26h en curso
db	0		
db	0		
Aux1	label	word	
dw	0		
Aux2	label	word	
dw	0		
ASSUME CS: CODE, DS: NOTHING			
Int21	proc	near	; CALL Int21 reemplaza a INT 21h
pushf			; Empuja flags para emular INT
call	cs:Vector21		; Control al vector original
ret			
Int21	endp		
Servicio21	proc	far	; Intercepcion de la int 21h
mov	cs:Aux1, ax		; Estas instrucciones son necesarias
mov	cs:Aux2, bx		; para que el programa que ejecuto
pop	ax		; la int 21h reciba los flags tal
pop	bx		; como DOS los dejo
popf			
push	bx		
push	ax		
mov	ax, cs:Aux1		
mov	bx, cs:Aux2		
mov	cs:Flag21, TRUE		; Pone Flag21
call	Int21		; Ejecuta Int 21
mov	cs:Flag21, FALSE		; Quita Flag21

Fig. 2 - Intercepción de DOS

talla, y programas de adquisición de datos son ejemplos típicos.

En todos ellos es necesario salvar el carácter no reentrante de DOS.

Flags

Un posible enfoque del proble-

ma es emplear ciertos **flags** de DOS que indican cuándo el operativo tiene el control; en teoría, basta abstenerse de invocar al operativo si éste ya está procesando un pedido de servicio.

En la práctica, esta posible so-

lución presenta dos graves inconvenientes.

El primero es que se descansa en aspectos de DOS sujetos a cambio *sin previo aviso*. Una nueva versión de operativo puede ser incompatible con la solución pretendida.

El segundo es que nada puede asegurar **que se tendrá oportunidad** de efectuar el acceso requerido: los intentos de acceso pueden estrellarse repetidamente contra un DOS ocupado.

A campo traviesa

En un mundo caótico, las soluciones no se basan en normas. En el problema que nos ocupa, lo mejor es interceptar -elegantemente- las funciones de DOS para saber cuando están en curso. A la vez nos desentendemos de la volubilidad de Microsoft, y establecemos un seguro de acceso.

La intercepción se basa en redirigir el vector de cada interrupción relevante de DOS hacia nuestro propio programa, el cual ejecutará el servicio original de DOS, pero antes pondrá un **flag** (cuyo significado es TRUE: servicio DOS en ejecución) y lo borrará al terminar. Todo intento de acceso deberá consultar estos flags para determinar si el acceso es posible. Una gran ventaja se obtiene a partir de una pequeña modificación: si el intento de acceso encuentra a DOS no disponible,

pone un flag indicando que hay una **operación pendiente**.

Nuestro programa monitorea los servicios de DOS; la posibilidad de que DOS esté libre se presenta (y se detecta tan rápido como es posible) cada vez que uno de ellos es completado. Basta verificar, luego de cada servicio, si ningún otro está en curso. Por motivos de economía de software, en la implementación ejemplificada sólo se hace esto si hay un acceso pendiente, y una rutina común se encarga de verificar si todos los flags de acceso a servicio están en off.

Esto significa que si un acceso no puede realizarse y queda pendiente, *la oportunidad de ejecutarlo se detectará inmediatamente*, y la demora será mínima.

Un ejemplo

Como el campo de aplicación de esta técnica es muy amplio, trataremos de ser cuidadosos en su ejemplificación. En su aspecto **específico**, este ejemplo sólo cumple un propósito: redirigir la salida de impresora a un archivo en disco. En este sentido, se trata de un programa incompleto, ya que serían deseables funciones no implementadas aquí (por ejemplo, disponer de hot keys para retornar el servicio de impresora a su estado normal, y abrir y cerrar diferentes archivos). El objetivo es, fundamentalmente, **ilustrar el**

```

call FileOps ; Verifica si hay escritura pendiente
ret ; retorna
Servicio21 endp
Servicio25 proc far ; Intercepcion de la int 25h
mov cs:Flag25, TRUE
call cs:Vector25 ; Ejecuta Int 25
mov cs:Flag25, FALSE
ret
Servicio25 endp
Servicio26 proc far ; Intercepcion de la int 26h
mov cs:Flag26, TRUE ; Pone Flag26
call cs:Vector26 ; Ejecuta Int 26
mov cs:Flag26, FALSE ; Quita Flag26
ret
Servicio26 endp
HookDOS proc near ; Rutina de intercepcion de
; interrupciones
pushf
cli
push ds
xor ax, ax ; DS en 0
mov ds, ax
mov bx, 21h * 4 ; Copia y sustituye vector 21h
mov ax, [bx]
mov cs:Vector21L, ax
mov [bx], offset Servicio21
mov ax, [bx+2]
mov cs:Vector21H, ax
mov [bx+2], cs
mov cs:Flag21, FALSE
mov bx, 25h * 4 ; Copia y sustituye vector 25h
mov ax, [bx]
mov cs:Vector25L, ax
mov [bx], offset Servicio25
mov ax, [bx+2]
mov cs:Vector25H, ax
mov [bx+2], cs
mov cs:Flag25, FALSE
mov bx, 26h * 4 ; Copia y sustituye vector 26h
mov ax, [bx]
mov cs:Vector26L, ax
mov [bx], offset Servicio26
mov ax, [bx+2]
mov cs:Vector26H, ax
mov [bx+2], cs
mov cs:Flag26, FALSE
pop ds
popf
ret
HookDOS endp

```

Fig. 2 - Continuación

acceso a archivos. Comencemos, pues, por analizar los diferentes módulos.

En primer término, durante la **instalación** del TSR, se interceptan las interrupciones 21h (servicios generales de DOS),

25h (lectura de sectores) y 26h (escritura de sectores). Las direcciones originales de cada vector se conservan; la intercepción ejecuta esos servicios, poniendo en ON sendos flags mientras el servicio está activo.


```

Handle    label    word    ; Handle del archivo abierto
          dw       0FFFFh
WriteAdd   label    word    ; Direcccion inicial a escribir
          dw       0
WriteLen   label    word    ; Longitud de datos a escribir
          dw       0

Request    label    byte    ; Flag de escritura pendiente
          db       0
ASSUME CS: CODE, DS: NOTHING

FileOps    proc     near
    pushf
    cli
    test    cs:Request, TRUE
    jz      FileOps_1
    test    cs:Flag21, TRUE
    jnz     FileOps_1
    test    cs:Flag25, TRUE
    jnz     FileOps_1
    test    cs:Flag26, TRUE
    jnz     FileOps_1
    push    ax
    push    bx
    push    cx
    push    dx
    push    si
    push    di
    push    ds
    push    es

    push    cs
    pop     ds

    ASSUME CS: CODE, DS: CODE
    mov     Request, 0
    call    WriteFile
    pop     es
    pop     ds
    pop     di
    pop     si
    pop     dx
    pop     cx
    pop     bx

    pop     ax
FileOps_1:
    popf
    ret
FileOps    endp

ASSUME CS: CODE, DS: CODE

FileName   label    byte    ; Nombre de archivo
          db       'C:\ARCHIVO.DAT', 0
WriteFile   proc     near    ; Escritura de archivo
    mov     ax, 3D41h        ; Apertura
    mov     dx, offset FileName
    call    int21
    jnc     WriteFile2

    mov     ah, 3Ch          ; Si falla, creacion
    mov     cx, 0
    mov     dx, offset FileName
    call    int21
    jc      WriteFile3

WriteFile2:
    mov     Handle, ax

    mov     bx, Handle        ; Posicionamiento
    mov     ax, 4202h        ; en fin de archivo
    mov     cx, 0
    mov     dx, 0
    call    int21
    jc      WriteFile3

    mov     ax, 4000h        ; Escritura
    mov     bx, Handle
    mov     cx, WriteLen
    mov     dx, WriteAdd
    call    int21
    jc      WriteFile3

    mov     ah, 3Eh          ; Cierre
    mov     bx, Handle
    call    int21

WriteFile3:
    ret
WriteFile   endp

```

Fig. 3 - Manejo del archivo

Más tarde veremos el papel del flag **Request** y de la ramificación asociada a él.

El procedimiento **Int21** llama al servicio original de DOS, y deberemos recurrir a él, en lugar de a la **Int 21h**, so pena de iniciar un *loop indefinido*.

El procedimiento **WriteFile** se

encarga del manejo de disco, empleando los servicios de DOS, a través del mencionado procedimiento **Int21**, para abrir, escribir y cerrar el archivo cada vez que resulta necesario.

Aquí cabe hacer notar que podría haber numerosas variantes. Otras aplicaciones pueden

requerir acceso para lectura, o para lectura/escritura, o una apertura diferente (el código **42h** en **AL** indica modo **READ/WRITE SHARED**). Asimismo, es posible que se necesite comandar la apertura y cierre mediante **hot keys**, o a través de ciertos eventos, y podría ser deseable suministrar externa-

BufferLen	EQU	1000h			call	FileOps
Buffer1	label byte				WriteBuffer3:	
db	BufferLen	dup(0)			pop	ds
Buffer2	label byte				pop	bx
db	BufferLen	dup(0)			pop	ax
BufferEnd	label word				ret	
BufferPtr	label word				WriteBuffer	endp
dw	0					
WriteBuffer	proc	near	; Escritura en buffer		FlushBuffer	proc near ; Vaciado de buffer
push	ax				push	bx
push	bx				push	ds
push	ds				push	cs
					pop	ds
push	cs				mov	bx, BufferPtr
pop	ds				sub	bx, offset Buffer1
					cmp	bx, BufferLen
mov	bx, BufferPtr				jae	FlushBuffer1
mov	[bx], al				mov	WriteLen, bx
inc	bx				mov	WriteAdd, offset Buffer1
cmp	bx, offset BufferEnd				jmp	FlushBuffer2
jb	WriteBuffer1				FlushBuffer1:	
					sub	bx, BufferLen
mov	BufferPtr, offset Buffer1				mov	WriteLen, bx
mov	WriteAdd, offset Buffer2				mov	WriteAdd, offset Buffer2
jmp	WriteBuffer2				FlushBuffer2:	
WriteBuffer1:					cmp	WriteLen, 0
mov	BufferPtr, bx				je	FlushBuffer3
cmp	bx, offset Buffer2				call	WriteFile
jne	WriteBuffer3				FlushBuffer3:	
WriteBuffer2:					pop	ds
mov	WriteLen, BufferLen				pop	bx
mov	Request, TRUE				ret	
					FlushBuffer	endp

Fig. 4 - Manejo de buffer

mente el nombre y ubicación del archivo.

Encolado (buffering)

Es sabido que nunca resultan conveniente los accesos que involucran unos pocos bytes; antes bien, es preferible manejar paquetes más extensos.

Por otro lado, dado que DOS no siempre estará disponible para una operación de este tipo, es necesario disponer de un almacenamiento temporal de los datos hasta que los mismos puedan ser escritos.

En este ejemplo se emplea un buffer circular **dividido en dos secciones**.

Cada vez que una de ellas se completa, los nuevos datos que arriban se almacenan en la restante; la primera es escrita a disco en la primera oportunidad disponible.

El procedimiento **WriteBuffer** se encarga de encolar apropiadamente cada byte de datos, codificado en el registro AL.

Como se prevé que la función del TSR pueda ser desactivada

(en el ejemplo, imprimiendo el caracter 26, fin de archivo), se suministra además un procedimiento de **vaciado del buffer**, que causa la escritura aunque la sección activa no esté completa.

En este ejemplo los datos derivados al buffer son tomados de la interceptación del servicio de impresora. Esta es la parte no genérica del ejemplo, y, desde luego, deberá ser sustituida por la deseada, por ejemplo, un módulo de adquisición de datos.



Desafío



El Último
Postulado
de Fermat.

La exploración más
extensa de conjuntos de
numeros naturales
verificando el postulado:

$$a^n + b^n = c^n \text{ falso si } n > 2$$

nos fue suministrada
por Héctor Hugo
Mazzeo, de La Plata,
quien remite un
programa que entiende
la verificación a

$$a, b \leq 600, n > 2.$$

Nuestras
congratulaciones para
Héctor, que desde ahora
se convierte en
beneficiario de una
suscripción anual
gratuita de PC Práctica.

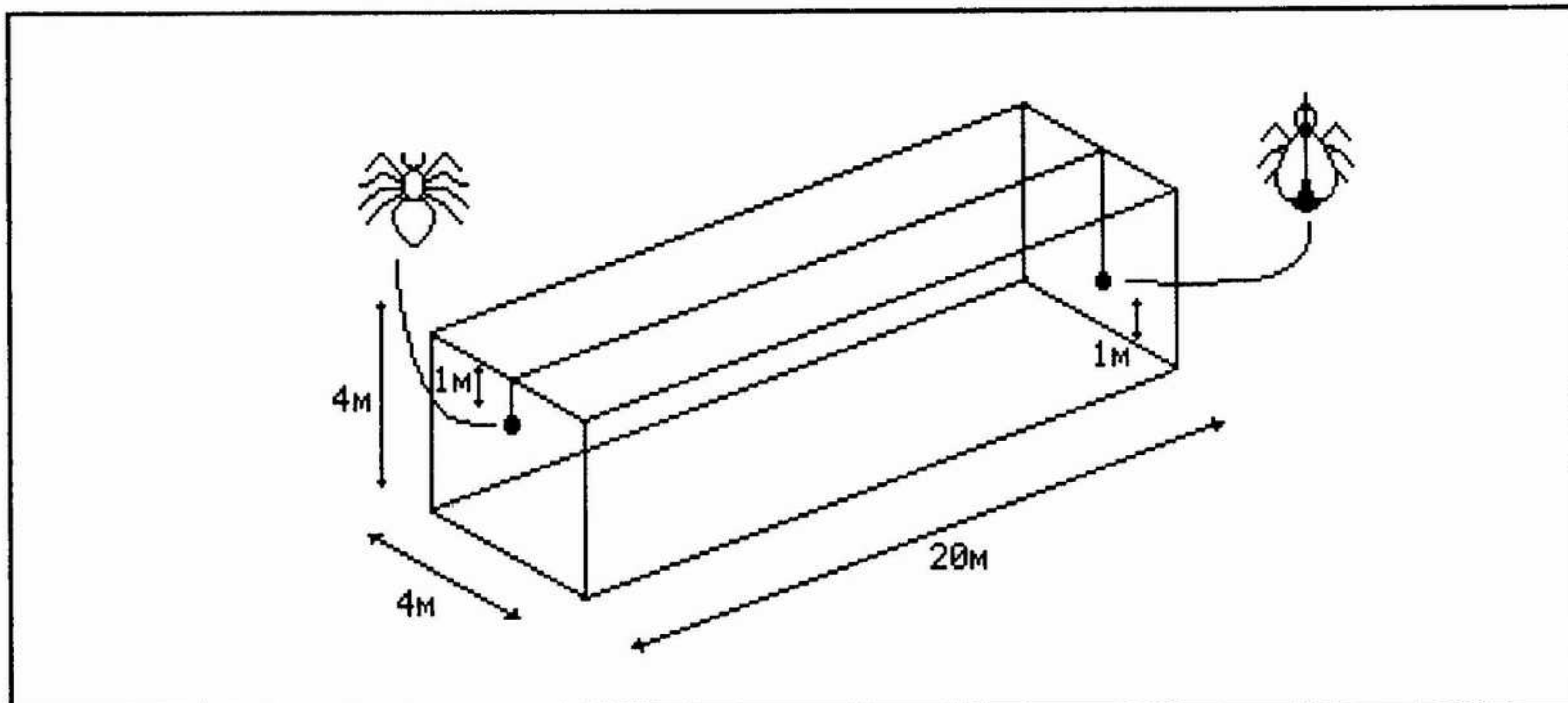
Nos cabe mencionar
además, la contribución

La araña y la mosca

¿Cuál es la línea más corta entre dos puntos? Sin duda, una recta. ¿O hay que dudar? Veámoslo: La línea más corta entre dos puntos se denomina **geodésica**, y en algunos casos es la recta que los une. La geometría no tiene obligación de que sus representaciones se parezcan a nada. Aun así, se suele considerar como "natural" la Euclídea (la tradicional), a pesar de que involucra conceptos que difícilmente hallan paralelo en nuestra experiencia cotidiana, tales como puntos y rectas ideales.

Pero tampoco la geometría Euclídea alcanza para describir el mundo, y para encontrar ejemplos de esto no hace falta enfrentarse a los "cucos" de las geometrías de Riemann o Lobatchevsky. Nuestro propio ámbito, el planeta, *no es un espacio (bidimensional) euclídeo*: se trata de la superficie de una cuasi esfera. En ella, los ángulos de un triángulo *siempre suman más de 180°*. Y las geodésicas, cuyo nombre proviene precisamente del hecho de que representan las distancias de navegación más cortas, son aproximadamente arcos de círculos máximos (es decir, centrados en el mismo centro del planeta).

Para ilustrar esto, que en un principio puede parecer parte de un razonamiento forzado, o de un acertijo con trampa, vamos a tomar un espacio muy simple: la superficie de un cuarto rectangular, de 20m de largo, 4 de alto y 4 de ancho.



El problema de la araña y la mosca

Pues bien, posada en el centro de una de las paredes cuadradas, y a 1m del suelo, se halla una mosca. En la pared opuesta, también en el centro, pero a 1m del techo, una araña.

La mosca ha accedido gentilmente a permanecer quieta mientras dura este ejercicio; con respecto a la araña, digamos que le agrada la geometría casi tanto como la degustación de moscas. Y su intención es llegar a su víctima lo antes posible, por lo que habrá de recorrer una geodésica en esta sencilla superficie.

La primera que se nos ocurre es la indicada en la figura, cuya longitud es de 24 m. Si el lector se aviene a investigar en este ejercicio, descubrirá *por qué la araña desdeña este camino*.

La respuesta que con mejor fundamento describa la ruta más corta entre ambos bichos será premiada con una suscripción gratuita de **PC Práctica**, recibándose las respuestas hasta el 20 de mayo. 🖱️

NOTA: Durante la realización de este artículo no se ha puesto en peligro la vida de ningún animal.

de Javier Valentini,
quien indica que el
Postulado puede
extenderse del
siguiente modo:

$$a_1^n + a_2^n + \dots + a_m^n = c^n$$

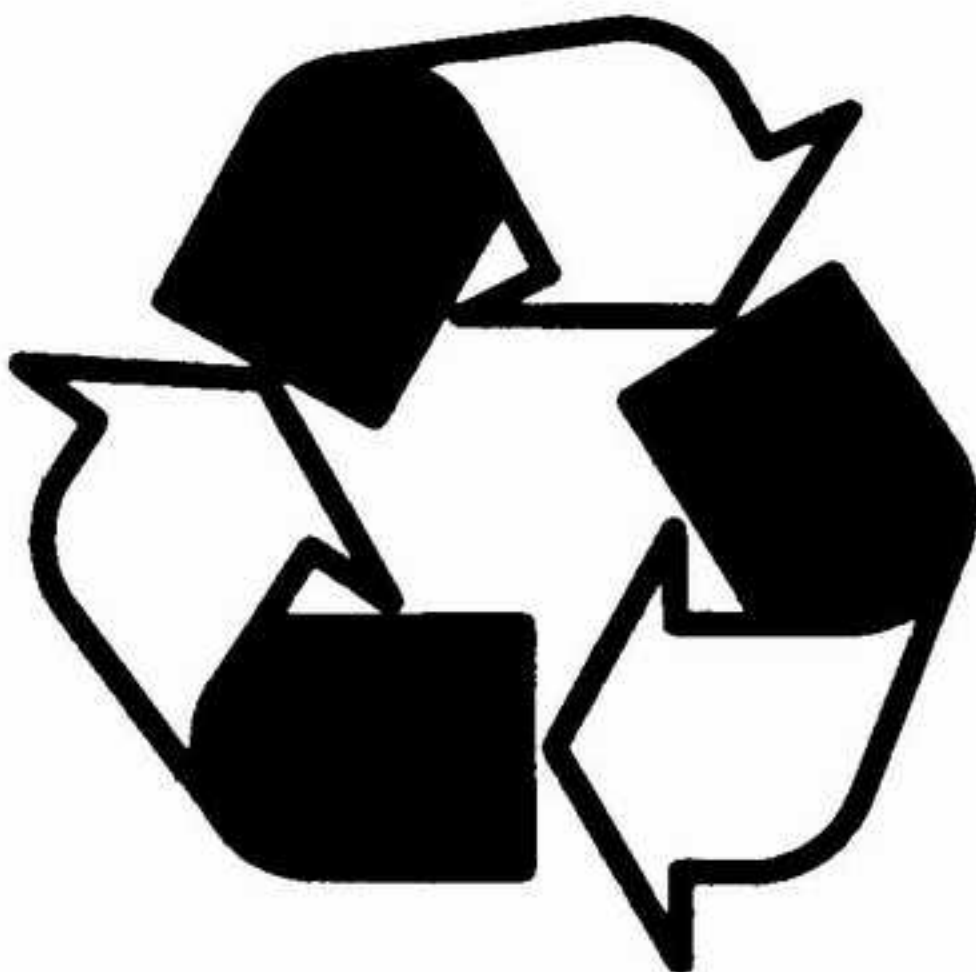
falso para todo $n > m$ □





El ahorro energético

La onda verde en la informática: aquí, por ahora, muy verde



En cada decenio se imponen modas. Las modas sólo se pueden juzgar pasada cierta edad (lo cual le otorga al juzgador un carácter algo lamentable). 'Fijemos esa edad en lo que guste a cada uno; pero es que una moda no se puede apreciar *si es la primera que uno aprecia*.

Una vez transpuesto ese límite, es inevitable seguirla, o coincidir con Wilde: *'Es una costumbre tan insoportable que hay que cambiarla cada dos años'*.

Actualmente la moda es la **ecología**, y no se piense que estoy a favor de la tala de árboles o de la extinción del lagarto overo. Lo que sucede es que cualquier actitud asumida por simple imitación suele aproximarse peligrosamente al disparate.

Existen serias razones para considerar muy importante la

preservación del medio ambiente, la mayoría de las cuales son simple consecuencia de haberlo destruido previamente. Por lo tanto, en muchos países se han adoptado medidas muy concretas, por una parte, y se ha creado una conciencia social muy fuerte al respecto.

El impacto publicitario

Una de las consecuencias de esto último es el fuerte impacto publicitario que resulta de cualquier mención a la preservación del medio ambiente, tenga o no asidero concreto.

Es bien conocido el efecto positivo de la exhibición del símbolo de **reciclado** en un envase. No obstante, el reciclado no es una solución en sí. Su efectividad depende de un sutil balance energético: si se gasta más energía en recolectar, transpor-

tar y reciclar que la necesaria para producir desde cero, el impacto sobre el medio ambiente será negativo, salvo que el material sea de disposición final realmente difícil o problemática.

Energía

Los argentinos estamos verdaderamente lejos del primer mundo que miramos con tanta devoción. En este aspecto, nos llega más lo superficial que lo esencial, lo que en este caso se traduce en que somos más propensos a mirar con buenos ojos el símbolo de reciclado que a pensar de dónde proviene su ventaja, del mismo modo que nos preocupa la extinción de la nutria más que la de los jubilados.

La electrónica verde

Detrás de las medidas destinadas a la preservación de la ecología, la generación de energía tiene un papel muy importante, que en la Argentina se desdibuja. El fundamento básico es que por cada kilowatt/hora consumido de más, se necesita generar un Kwh extra, *lo cual incide sobre el medio ambiente.*

Porque la generación de ese Kwh demandará quemar combustible, talar árboles para construir represas, o instalar centrales nucleares, que aunque fuesen perfectamente seguras no dejarían de alterar el medio, al elevar la temperatura de las vías de agua cercanas

con sus circuitos de enfriamiento.

El resultado primario es que toda reducción de consumo de energía futuro redundará en un alivio de la degradación del medio ambiente. En los países del hemisferio norte, el parque de computadoras instaladas y su previsible crecimiento es lo suficientemente importante -en términos de consumo- como para alentar **técnicas de ahorro de energía.**

Así es que se promocionan actualmente motherboards verdes, monitores verdes y una legión de equipamiento de bajo consumo.

¿Y aquí?

En Argentina, los términos de esta relación aparecen muy cambiados. El impacto ambiental de un incremento en el parque de PCs es pequeño, no sólo por el tamaño de ese parque, sino porque el problema energético *es bastante diferente:*

En materia de generación estamos bastante atrasados, cualitativa y cuantitativamente. A principio de los 70's se calculaba que aún aprovechando todos los recursos hidroeléctricos, se iba a requerir la construcción de una central como Atucha cada cuatro años para poder arribar al 2000 sin déficit energético.

Esto, por supuesto, no se hizo, y Yaciretá recién entró en servicio el año pasado (luego de ba-

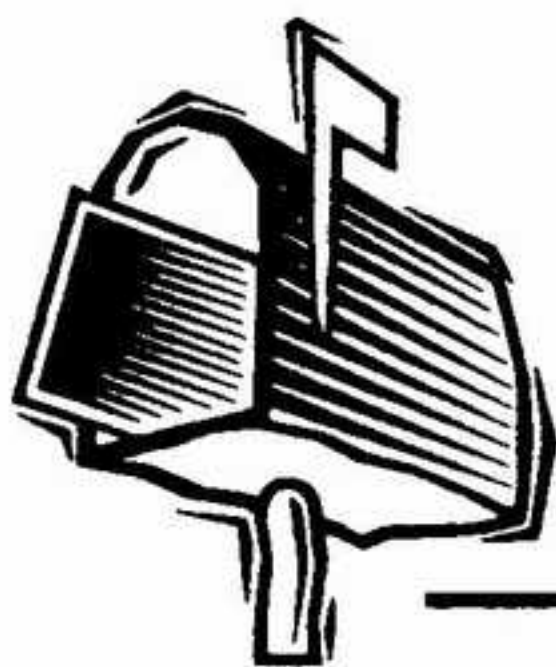
tir todos los records de atrasos, coimas, y concesiones varias). Un atisbo del déficit previsto lo tuvimos hace seis años, cuando un verano particularmente seco obligó a programar severos cortes de suministro.

La realidad es que el déficit esperado no se manifestó en plenitud por una sola razón: la cantidad de industrias con las cortinas bajas. Las previsiones de demanda energética contaban con una industria creciente, por lo menos en función de su comportamiento histórico.

Si se trata de hacer números, hay dos caminos. El primero -el empleado uniformemente por las entidades gubernamentales de todos los tiempos- es acudir a estadísticas que son sumas de pequeñas mentiras (*por ejemplo, el sueldo de un ñoqui aparecerá incrementando el Producto Bruto Nacional*). La alternativa es acudir a indicadores menos maleables, como el consumo de acero, de cemento, de ácido sulfúrico ... o de energía.

Concluyendo

La electrónica ecológica tiene un sentido que es función de su entorno: preservación indirecta del medio ambiente por medio del ahorro de energía; en nuestro país, y a la fecha, no tiene mucho más sentido que el de un atractivo slogan.



Correo de lectores

Cuento chino

Sr. Director:

.... Al margen, me interesaría plantear un par de consultas.

La primera se refiere al empleo de sensores de efecto Hall, que por lo visto son bastante difíciles de conseguir (los pocos que he encontrado son lineales). ¿Hay alguna forma de utilizarlos como sensores digitales?

La segunda pregunta es más bien una curiosidad. Leí en alguna parte, creo que en un folleto, una recomendación acerca de que en casos de problemas con el color se indicaba dejar apagado el monitor por lo menos media hora. ¿Esto es un disparate taiwanés o tiene algún fundamento?

... Agradeciendo desde ya su atención lo saludo atentamente.

*Claudio Aragón
Capital Federal*

PCP: Ud. puede emplear un sensor como los mencionados (p. ej. UGN3503), siempre que acondicione y discretice la señal, que corresponde a una tensión de alrededor de 2.5V, y cuya sensibilidad es bastante baja. Esta señal se debe amplificar (referida a una tensión fija de aproximadamente 2.5V), y la salida se puede conducir a una compuerta Schmitt (74LS14) para obtener el grado de histéresis adecuado. Una forma de prescindir de un operacional con fuente partida

es emplear un transistor con un resistor de 50 a 100K y tres diodos (p. ej. 1N4148) en serie con la base, sin polarización adicional (la caída de tensión en las cuatro junturas se aproxima a 2.2-2.4V). Con esto, y una resistencia de colector de alrededor de 1K, se obtiene una ganancia de 5-10.

Con respecto a los monitores, no se trata de un *cuento chino*. Que cada haz incida sobre el fósforo del color correspondiente se consigue mediante una máscara y un enfoque -magnético- muy cuidadoso. De hecho, el monitor incluye imanes permanentes de compensación.

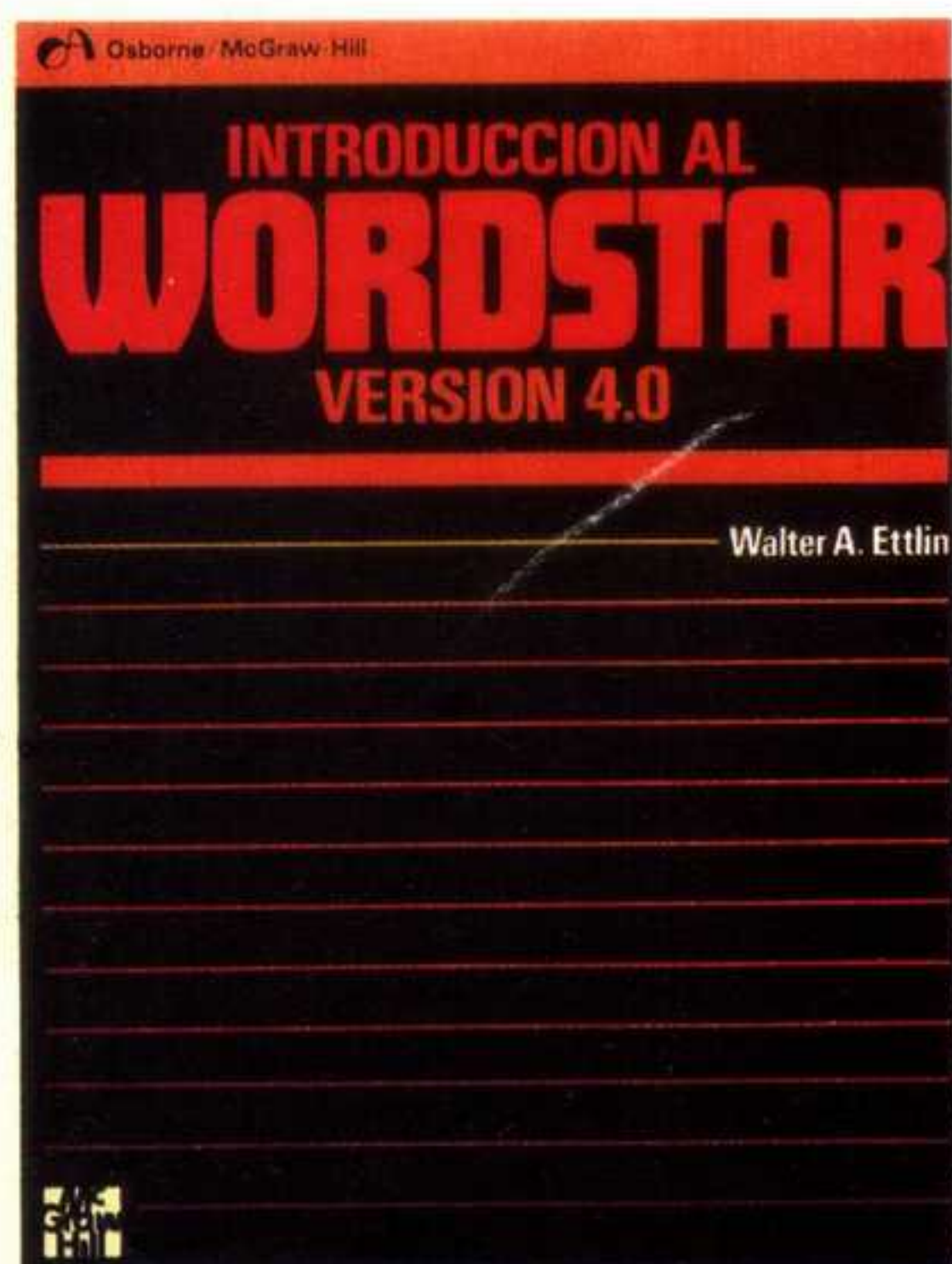
Si este campo magnético cuidadosamente calibrado se altera, cada haz caerá, más o menos al azar, sobre puntos del color equivocado, y el resultado será un viraje del color que puede ir desde un leve irisamiento a un carnaval carloca.

Una de las causas de este problema -al que también están expuestos los equipos de TV es la magnetización de la propia máscara, y para suprimirlos muchos monitores cuentan con una bobina de desmagnetización (degaussing), que genera un campo magnético alternativo, inicialmente intenso, y paulatinamente decreciente.

Como el degaussing consume bastante potencia, se suele incorporar un temporizador que lo pone en marcha sólo cuando el monitor se enciende después de un periodo de apagado que, como Ud. menciona, suele ser de alrededor de media hora.

Conocer esta característica es también importante cuando se miden potencias consumidas: el consumo es sensiblemente superior en las condiciones citadas.

...y en procesadores de texto:



Introducción al WordStar v.4.0

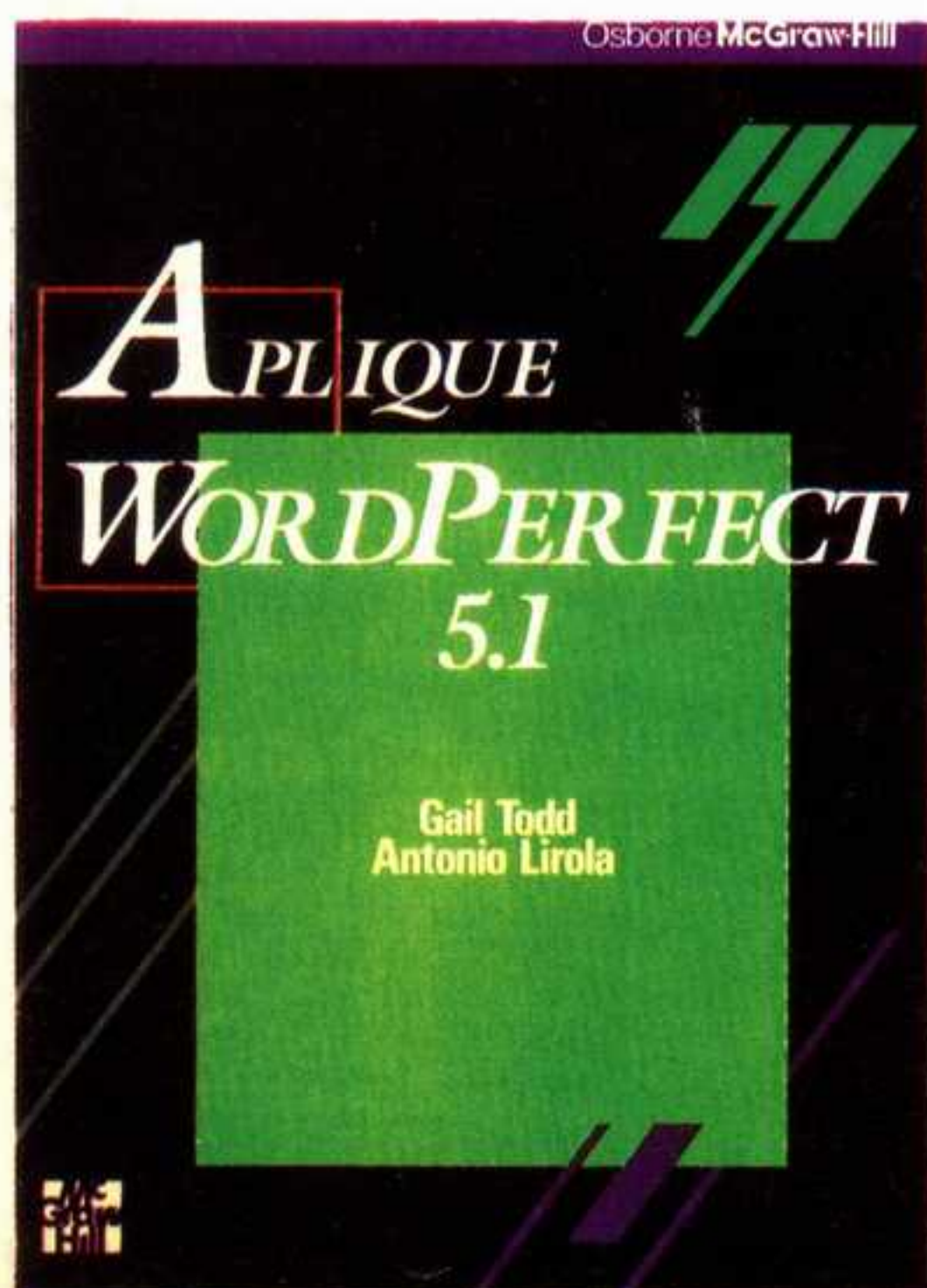
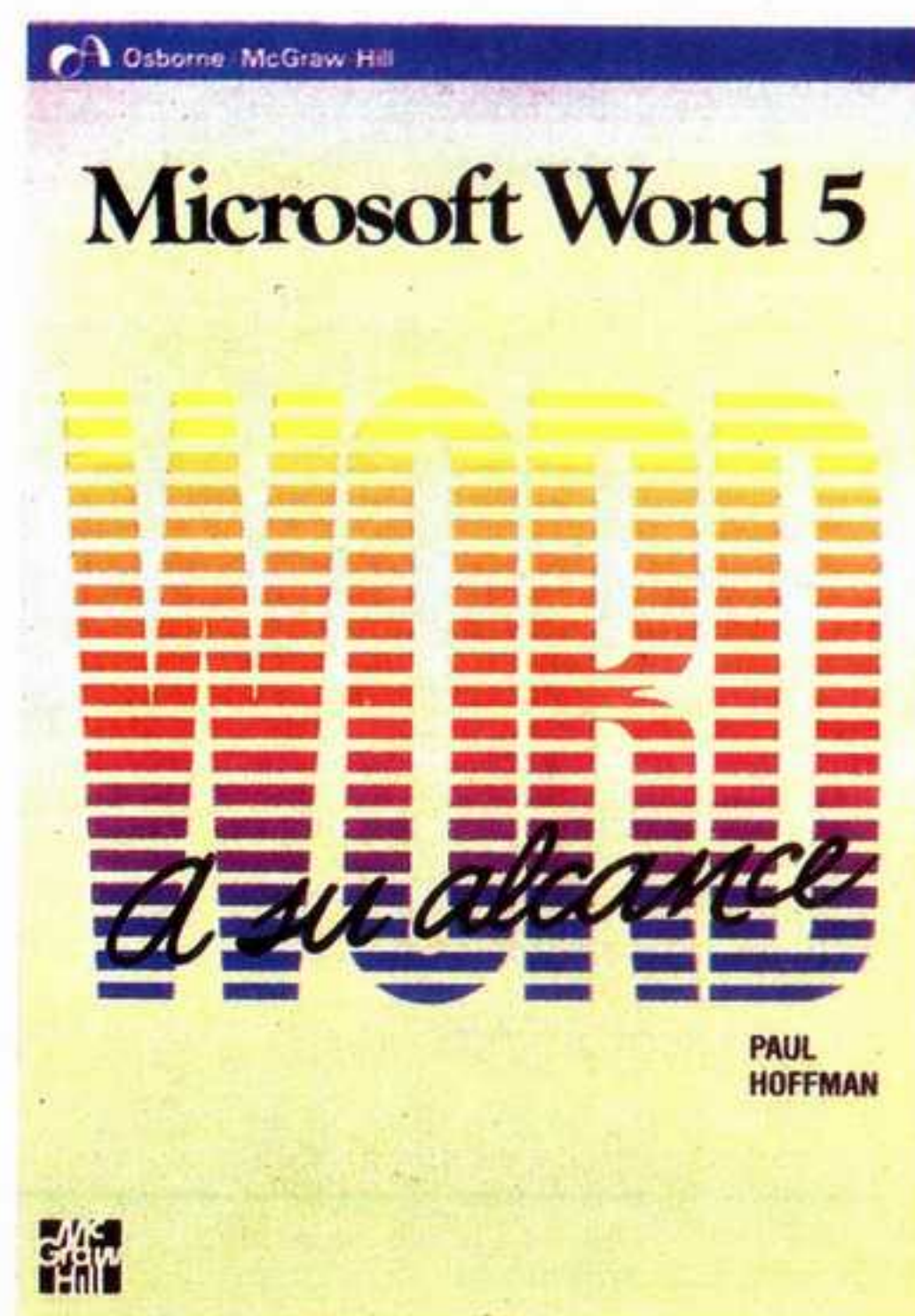
El propósito de este libro es ayudarle a llegar a utilizar eficientemente el WordStar, un programa de tratamiento de textos que brinda óptimas posibilidades de creación, búsqueda de palabras, corrección o impresión de documentos de cualquier tipo.

\$ 29.-

MS Word 5 a su alcance

Microsoft Word 5 a su alcance introduce al lector en este potente programa de procesamiento de textos, guiándolo paso a paso y reforzando el aprendizaje con numerosos ejemplos prácticos de edición, formateo y almacenamiento de textos.

\$ 29.-



Aplique WordPerfect 5.1

WordPerfect es un procesador de textos avanzado, con el que se podrán llevar a cabo desde las más simples hasta las más complejas operaciones de tratamiento de textos. Este libro lo habilitará para utilizar WordPerfect 5.0 y 5.1 con destreza y eficacia.

\$ 40.-

Solicítelos agregando \$ 6.- para envío por Correo Certificado, mediante giro postal o bancario a la orden de:

EDITORIAL CUL-TEC S.A.

Independencia 1654 - 1100 Buenos Aires

Tel/Fax 383-7126 381-9308 381-9327

Horario: Lunes a Viernes de 10 a 17 horas

**Un clásico de todos los viernes
desde hace
45 años**

**Siempre
actualizada,**

**con todo el
aporte de la
computación
a la
electrónica**



**En
venta
en todo
el país.
Solicítela
en su kiosco,**

**Editorial
Cul-Tec S.A.**